Music-Mike

Harvey Wu, Kaitlin Pet, Lakshmi Bodapati, Husam Abdul-Kafi

May 11, 2017
Contents

1 Introduction 5
  1.1 Design Ethos .............................................. 6

2 Language Tutorial 7

3 Language Reference Manual 9
  3.1 Types, Operators, and Expressions ...................... 9
    3.1.1 Basic Types ............................................. 9
  3.2 Expressions ............................................... 11
  3.3 Variables and Assignment ............................... 13
  3.4 Operators ................................................. 14
    3.4.1 Arithmetic Operators ................................. 14
    3.4.2 Logical and Comparison Operators .................... 15
    3.4.3 Pitch Operators ........................................ 15
    3.4.4 List Operators ......................................... 15
  3.5 Control Flow .............................................. 16
    3.5.1 Statements ............................................. 16
3.5.2 Expressions ................................................. 16
3.5.3 If-Then-Else ................................................. 16
3.5.4 Block .......................................................... 16
3.6 Functions and Program Structure .............................. 17
  3.6.1 Functions ................................................... 17
  3.6.2 Built-in Functions ........................................... 17
3.7 Comments ........................................................ 18
3.8 Scoping ............................................................ 18
3.9 Built in Functions ............................................... 18
  3.9.1 Printint ....................................................... 18
  3.9.2 Printstr ....................................................... 18
  3.9.3 Printfloat ..................................................... 18
  3.9.4 Printlist ....................................................... 19
  3.9.5 Printrlist ...................................................... 19
  3.9.6 Synth .......................................................... 19
  3.9.7 Merge ........................................................ 19
  3.9.8 Make_midi ..................................................... 19

4 Project Plan ...................................................... 20
  4.1 Process .......................................................... 20
    4.1.1 Planning .................................................... 20
    4.1.2 Specifications ............................................. 21
    4.1.3 Development ............................................... 21
4.1.4 Testing .................................................. 22
4.2 Programming Style Guide .......................... 22
4.3 Timeline .................................................. 23
  4.3.1 Roles and Responsibilities ....................... 24
  4.3.2 Software Development Environment ............... 24
  4.3.3 Project Log ........................................ 25

5 Architectural Design ................................. 46
  5.1 Block Diagram ....................................... 46
  5.2 Interfaces ............................................. 48
    5.2.1 Scanner (Lakshmi, Husam, Harvey) ............. 48
    5.2.2 Parser (Kaitlin, Husam, Harvey, Lakshmi) ....... 48
    5.2.3 Type Inference (Harvey) ........................ 49
    5.2.4 Semant (Lakshmi, Husam, Harvey) ............... 49
    5.2.5 The Code Generator (Husam, Kaitlin, Lakshmi) .. 50

6 Test Plan ............................................... 51
  6.1 Example tests ....................................... 51
  6.2 Test Suite ........................................... 126
  6.3 Test Suites .......................................... 138
  6.4 Test Automation ..................................... 138

7 Lessons Learned ........................................ 139
  7.1 Team Reflections .................................... 139
7.1.1 Harvey Wu ........................................... 139
7.1.2 Kaitlin Pet .......................................... 140
7.1.3 Lakshmi Bodapati ................................. 140
7.1.4 Husam Abdul-Kafi ................................. 141

8 Appendix ................................................. 143
1. **Introduction**

Western music is usually notated on a five-line staff, on which *notes* are given a *duration* based on symbol type, and *pitch* based on location in the staff. Composers can use proprietary software such as Sibelius or Finale to manipulate a virtual five-line staff through mouse clicks or keyboard gestures. Fans of computer music might instead use music synthesis libraries to programmatically create music in languages such as C++, but such libraries can be unintuitive for musicians unfamiliar with signals and waves.

We propose Music-mike, a compiled, strongly typed, programming language to give users an alternative option in music creation. Music-mike is designed for users to create music based on varied manipulations of short patterns. We owe this idea to Note Hashtag, a previous project completed in COMS W4115. However, unlike Note Hashtag, Music-mike is *modal* rather than *key-based*. Furthermore, lists - treated as the fundamental building block of music - are manipulated with special list operators (syntactic sugar) which create an intuitive interface based on traditional staff notation.
1.1 Design Ethos

The most basic unit in music is a note, which can be decomposed into pitch and duration. A simple melody can thus be described as two lists: one list of pitches and another of durations. A chord is a collection of notes played simultaneously.

A mapping of pitches is defined as a mode. All modes are subsets of the chromatic scale, which contains all twelve pitch classes used in Western music. Most music constrains the pitches of its notes to a small set of familiar modes, such as the major and minor scales. The sound of a chord is very much dependent on the mode that its notes come from.

Music-mike is based on the following observations regarding Western music: one, that Western music is fundamentally chordal and modal. Two, that Western music is repetitive and manipulative: simple building blocks of music are modified, then repeated multiple times in a piece. Finally, and most importantly, that these simple building blocks can described using lists and altered using a functional paradigm.
2. **Language Tutorial**

**Basics**

Let’s try defining a variable first:

\[ x = 5; \]

Now, let’s write a function:

```plaintext
def AddFive a = a + 5;
```

Notice that we don’t need to add any type annotations. Type inference handles everything for us. We can also write polymorphic functions, like the identity function:

```plaintext
def Identity x = x; y = Identity("Who am I"); Printstr(y);
```

Function calls require parentheses around arguments, but no commas to separate. For example:

```plaintext
def Add j k = j + k; sum = Add(4 5);
```

Note that arithmetic operators do not overload: we cannot pass in an floats to our Add function:

```plaintext
/* This fails! */ wrong = Add(4.5 3);
```

**A Musical Hello World**

First we need to define a mode:
/ A major scale */ major = [1 3 5 6 8 10 12 13];

A rhythm list is defined with the r:[] constructor:

r1 = r:[s s s s s e s s s s s e];

A pitch list is defined with the p:[] constructor:

p1 = p:[1 1 5 5 6 6 5 4 4 3 3 2 2 1];

Now, we can generate a music string with the Synth function, which takes a pitch list, rhythm list, mode, start node, and channel:

startnote = 50;

s = Synth(p1 r1 major startnote 1); /* Make_midi outputs a midi file */ Make_midi(s "twinkle.midi");

And that is twinkle twinkle little star!

3.1 Types, Operators, and Expressions

All types are immutable in Music-Mike.

3.1.1 Basic Types

- Unit (unit)
  
The only value that unit can take is ()

- Boolean (bool)
  
  Takes two values: true or false.

- Integer (int)
  
  A 32-bit signed integer.

- Float (float)
  
  A 64-bit floating point number - follows the specifications of IEEE 754. Must contain a decimal point and either an integer or fractional component. The missing component is treated as a zero.
In the context of a rhythm list, floats can also be one of 6 characters that get scanned in as floats.

<table>
<thead>
<tr>
<th>char</th>
<th>float</th>
</tr>
</thead>
<tbody>
<tr>
<td>q</td>
<td>1.0</td>
</tr>
<tr>
<td>w</td>
<td>4.0</td>
</tr>
<tr>
<td>h</td>
<td>2.0</td>
</tr>
<tr>
<td>t</td>
<td>0.33</td>
</tr>
<tr>
<td>e</td>
<td>0.5</td>
</tr>
<tr>
<td>s</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Examples: 5. 6.43 3.1415 .42 q e

- **String (string)**

A simple string enclosed by double quotations not spanning multiple lines.

Examples: "hello" "music mike looks good AND sounds good"

- **Integer List**

A list of 32-bit signed integers surrounded by square brackets [ ], delimited by spaces.

Examples: [1 2 3 4] [57 0 65]

- **Rhythm List**

A list of 64-bit floating numbers and float characters (q, w, h, t, e, s) surrounded to the left by r: [ and to the right by ] and de-
limited by spaces. A Rhythm List is used to denote the length of each pitch/chord at each corresponding position in a Chord List.

Examples: \(r: [0.5 \ 6 \ 1.9 \ 37.0]\) \(r: [q \ w \ s \ t] \) \(r: [0.75 \ e \ s \ .09]\)

- **Chord List**

  A list of chords. A chord cannot appear anywhere else. A chord is a list of one or more pitches delimited by a vertical bar \(|\). A pitch is an integer pre-operated by zero or more \(\wedge\) and \(v\) s or post-operated by zero or more \(b\) or \(#\). The chord list is delimited by spaces and surrounded to the right by \(p:\) [ and to the left by \(]\). A \(\wedge\) semantically represents an Octave Up. A \(v\) semantically represents an Octave Down. A \(b\) semantically represents a Flat. A \(#\) semantically represents a Sharp.

Examples:

\(p: [1 \ 2 \ 3]\)

\(p: [1|3|5 \ 5 \ 6 \ 7]\)

\(p: [\wedge1|4|\wedge2# \ 5 \ v6|7|8###b## \ ]\)

### 3.2 Expressions

All expressions have return values in Music-Mike. An expression could be:

1. An integer, float or boolean literal constant:
2. A string constant enclosed by quotation marks

“string”

3. An arithmetic operation:

\[ e_1 \text{ op } e_2 \]

4. An if – then – else statement:

\[ \text{if} \ e_1 \ \text{then} \ e_2 \ \text{else} \ e_3 \]

5. Variable declarations and assignment:

\[ x = e_1 \]

6. Variables:

\[ x \]

7. A block of expressions. The last expression \((e_n)\) is the value of the block:

\[ \{e_1; e_2; \ldots e_{n-1}; e_n\} \]

8. A Function Declarations. The name of the function starts with a Capital letter. The number of arguments \(n\) is greater than or equal to 0.

\[ \text{def Fname } arg_1 \ldots arg_n = e \]
9. Function call. The arguments are delimited by whitespace.

\[ \text{Fname} \ (e_1 \ldots e_n) \]

10. A white-space separated list of integers:

\[ [\text{int}_1 \ \ldots \ \text{int}_n] \]

11. A white-space separated list of floats and/or float characters:

\[ r : [\text{float}_1 \ \ldots \ \text{float}_n] \]

12. A white-space separated list of chords

\[ p : [\text{chord}_1 \ \ldots \ \text{chord}_n] \]

13. A concatenation of two list expressions:

\[ e_1 @ e_2 \]

14. Subsetting a list:

\[ e.\{\text{int}\} \]

### 3.3 Variables and Assignment

Variable Identifiers in Music-Mike are strings that can be expressed using the regular expression:

\[
[ 'a' 'c'-'u' 'w'-'z' ] | [ 'a' 'c'-'u' 'w'-'z' ]

\[
[ 'a'-'z' 'A'-'Z' '0'- '9' ' ' ] *
\]

Essentially, Variable Identifiers cannot start with a \( b \) or \( v \).
Function Identifiers in Music-Mike are strings that can be expressed using the regular expression:

\[
['A'-'Z'] | ['A'-'Z'] ['a'-'z' 'A'-'Z' '0'-'9''_
\]

Essentially, Function Identifiers have to start with an Uppercase letter.

There are no type annotations in Music-mike due to use of the Hindley-Milner type system. We can assign a value to a variable using the following syntax:

\[
identifier = expr
\]

and type-inference will figure out the type. Note that the assignment operator is non-associative.

### 3.4 Operators

#### 3.4.1 Arithmetic Operators

Binary arithmetic operators are strongly-typed; both operands must be of the same type, and use the correct operator for their type.

Binary integer operators in order of precedence: \(*\, /\, +\, -\, .\)

Binary float operators in order of precedence: \(*\, /\, +\, -\).
3.4.2 Logical and Comparison Operators

Comparison operators support integers and floats; both operands must be of the same type:

\(<\>\equiv!=

The following boolean comparison operators are listed in order of precedence:

\(\equiv!=&&||\)

3.4.3 Pitch Operators

All pitch operators are unary. The postfix operators ‘#’ ‘♭’ raise or lower pitch by a half step. The prefix operators ‘ˆ’ and ‘♭’ increment/decrement the octave of pitch by one.

3.4.4 List Operators

- Concatenation- One list followed by another of same type connected by an ‘@’ symbol.

- Index- gets value of element of list using .[] operator

\[
def \text{Get second } x = x[1]
\]
3.5 Control Flow

3.5.1 Statements

A statement is an expression followed by a semicolon.

3.5.2 Expressions

Expressions always have a return value. A constant expression returns its literal value, a variable expression returns the value in that variable. All of the list expressions return the list. Each of the expressions in a sequence of expressions has a value. Function declarations return the function itself as a value.

3.5.3 If-Then-Else

If statements are structured as `if boolean-condition then expr else expr`. If-Then-Else statements are themselves expressions and thus have return types. The expressions after `then` and `else` must have the same type.

```latex
fun iszero x = if x == 0 then true else false
```

3.5.4 Block

Blocks of code consist of semicolon delimited expressions and are enclosed by brackets ended by a semicolon.
3.6 Functions and Program Structure

3.6.1 Functions

Functions can be defined using the keyword `def`:

```
def Name arg1 ...argN = expr
```

Note that the first character of a function’s name has to be capitalized. Here’s an example:

```
def Plusfive x = x + 5;
```

We can also define our function to return more complex expressions:

```
fun Iszero x =
if x == 0 then true
else false
```

There is no function overloading in Music Mike. Declaring a different function of the same name is not legal. `Iszero 5.0` is not valid.

Functions are almost first-class citizens: they can be passed in as arguments to other functions and returned by functions, but user-defined functions cannot be nested. Thus we avoid the funarg problem and handling closure.

3.6.2 Built-in Functions

`Printint Printstr Printfloat Printlist Printrlist`  
`Synth Make_midi Merge`
3.7 Comments

Comments are enclosed by \* *\ There is no special single-line comment syntax, and nested comments are not supported.

3.8 Scoping

Once a variable has been defined, it cannot be redefined. All variables are stored in a global symbol table. In this sense Music-Mike is dynamically scoped.

3.9 Built in Functions

3.9.1 Printint

Given an integer value, prints it to standard output.

3.9.2 Printstr

Given a string value, prints it to standard output.

3.9.3 Printfloat

Given a float value, prints it to standard output.
3.9.4 Printlist

Given an integer list, prints it to standard output.

3.9.5 Printrlist

Given a rhythm list, prints it to standard output.

3.9.6 Synth

Takes a chord list, rhythm list, integer list (mode), integer starting note, and integer channel number, returns a CFugue string representation.

3.9.7 Merge

Takes two strings and returns them concatenated together.

3.9.8 Make_midi

Takes a CFugue string representation and a filename, generates the Midi file represented by that string and saves it.
4. Project Plan

4.1 Process

4.1.1 Planning

Our team met regularly twice a week on Wednesdays to meet with our TA Jacob Graff and on Sundays to work together as a team, debrief and set the course for the rest of the week. We used our Wednesday meetings as an opportunity to track and gauge our progress and also ask questions about difficult problems we came across during the previous week. We also talked about goals and milestones for the next Wednesday meeting and talked about any potential problems related to the difficulties of the goals we defined but also about any foreseeable road blocks related to tests, projects, other classes etc. that might hinder our progress. We used these meetings to make sure our project was progressing but also to shift our timeline to account for future roadblocks and delays.
4.1.2 Specifications

We spent the first three weeks deciding the specifications of our language. We all met near a piano either in the basement of the dorms or in Lerner and went over intuitive ways for musicians to express music. Once we chose how we wanted to abstract notes, pitches, chords, tempos and more, we started talking about how to structure our language. We initially chose to do a functional language modelled after OCaml, but as we progressed, we realized that for the use cases we were targeting a fully functional language wouldn’t give us the kind of ease of use and usage we’d like. Our first concrete specifications were the abstractions and then we decided on syntax. Despite having a very concrete definition of specifications early on, we still changed specifications as we worked on our language when it was necessary to be able to finish within our timeline.

4.1.3 Development

Our team used github issues to define specifications and tasks that needed to be implemented or completed. We used github to help with organizing our development. Each of used a separate branch to develop the feature that we were working on and then submitted a pull request to the main branch once we thought it was ready. Then, another member of the team would review that request and merge the request. This ensured that all the code we pushed had been code reviewed and helped us maintain the quality of our mainline
4.1.4 Testing

We developed a test suite that tested individual components of the compiler. Every time one of us was working on a small component, we first wrote a test for how that component was supposed to work once it was finished. When writing In this regard, we used some principles of Test-First Programming to make sure we were preserving the functionality of the older features but also ensuring functionality of the new ones. While some of these tests were forward looking and failed early-on, the error messages told the developer whether we were making progress towards making these larger full stack tests work or if it was failing in whatever module the developer was working on.

4.2 Programming Style Guide

- Landin’s pseudo law: Treat the indentation of your programs as if it determines the meaning of your programs. Keep indentation consistent with that of the MicroC code.

- Keep lines shorter than 80 characters.

- A function should always fit within one screenful (of about 70 lines), or in exceptional cases two, at the very most three. To go beyond this is unreasonable.
Justification: When a function goes beyond one screenful, it’s time to divide it into subproblems and handle them independently. Beyond a screenful, one gets lost in the code. The indentation is not readable and is difficult to keep correct.

- The change in indentation between successive lines of the program is 2 spaces.

- Using the tab character (ASCII character 9) is absolutely not recommended. Change your .vimrc if you have tabs.

- Use underscores instead of Camel case

### 4.3 Timeline

<table>
<thead>
<tr>
<th>Date</th>
<th>Milestone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 29th</td>
<td>First commit to repository</td>
</tr>
<tr>
<td>Feb 8th</td>
<td>Project Proposal and White Paper Completed and Submitted</td>
</tr>
<tr>
<td>Feb 22</td>
<td>Language Reference Manual Completed and Submitted</td>
</tr>
<tr>
<td>Mar 21</td>
<td>Basic Scanner Complete</td>
</tr>
<tr>
<td>Mar 26</td>
<td>Basic AST and Parser Complete</td>
</tr>
<tr>
<td>Mar 29</td>
<td>Hello World runs</td>
</tr>
<tr>
<td>Apr 7</td>
<td>Testing Framework Complete</td>
</tr>
<tr>
<td>Apr 23</td>
<td>Final Scanner, Parser and AST Complete</td>
</tr>
</tbody>
</table>
4.3.1 Roles and Responsibilities

While we had defined project roles at the beginning of the semester, about three weeks in the roles became a lot more fluid. Our assigned roles were Tester, System Architect, Project Manager and Language Guru. Each member was involved in developing certain functionalities and portions of components. The team frequently worked together either in-person or teleconference.

<table>
<thead>
<tr>
<th>Team Member</th>
<th>Role</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lakshmi Bodapati</td>
<td>Project Manager</td>
<td>Compiler Front-end, Documentation, Polymorphic Function Typing</td>
</tr>
<tr>
<td>Kaitlin Pet</td>
<td>Tester</td>
<td>Pitch and Chord Full Stack Abstraction, Library Linking, Testing Architecture</td>
</tr>
<tr>
<td>Harvey Wu</td>
<td>Language Guru</td>
<td>Type Inference, Compiler Front-end</td>
</tr>
</tbody>
</table>

4.3.2 Software Development Environment

1. Version Control: Git, Github

2. Languages: OCaml 4.04.0, C, Bash

3. Text Editor: Vim, Sublime Text, Atom

5. Virtual Machine: Google Cloud

4.3.3 Project Log
c400293 was Harvey Wu, 35 minutes ago, message: Added stuff to Final Report
9f98afa was Harvey Wu, 44 minutes ago, message: Added fail on redefining stdlib functions
03489fb was Harvey Wu, 69 minutes ago, message: And more cleanup
447b852 was Harvey Wu, 70 minutes ago, message: More cleanup.
356bda2 was Harvey Wu, 2 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
dc595a1 was Harvey Wu, 2 hours ago, message: Lots of cleanup. Style issues. More tests.
db9c39d was Mounika, 2 hours ago, message: Merge pull request #78 from wuharvey/documentation
98c5e08 was Lakshmi Bodapati, 2 hours ago, message: completed first draft. Needs to be latexed and Appendix filled out and personal reflections filled out and codegen section. Other sections either need diagrams or examples. LRM needs to be updated and made a part of this document
9945ce8 was habdulkafi, 2 hours ago, message: added test script
23c2de9 was Lakshmi Bodapati, 2 hours ago, message: added in more details about modules
4915533 was Lakshmi Bodapati, 3 hours ago, message: presentation ppt and pdf
c62f155 was Harvey Wu, 4 hours ago, message: Important stuff for presentation
84101fc was Harvey Wu, 4 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
4973127 was kpet123, 5 hours ago, message: Merge pull request #77 from wuharvey/husam-synth-new
99acf48 was Harvey Wu, 5 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
1f9059e was Harvey Wu, 5 hours ago, message: Fixed semant
ef13d4f was kpet, 5 hours ago, message: yay!
b0eddb37 was habdulkafi, 5 hours ago, message: stuff
ecd6017 was Husam Abdul-Kafi, 6 hours ago, message: Merge pull request #76 from wuharvey/synth-2
75725c1 was Husam Abdul-Kafi, 6 hours ago, message: Merge branch 'master' into synth-2
c11f8db was kpet, 6 hours ago, message: fixed make midi
8fa796c was kpet, 6 hours ago, message: Merge branch 'synth-2' of https://github.com/wuharvey/music-mike into synth-2

e867c0d was kpet, 6 hours ago, message: test worksgit add

b02e6de was habdulkafi, 6 hours ago, message: updated cfugue
exe w/ cmd line args

f90fd79 was Husam Abdul-Kafi, 7 hours ago, message: Merge pull request #75 from wuharvey/husam-strcat

e37d735 was habdulkafi, 7 hours ago, message: added string concatenation

47d4f11 was Husam Abdul-Kafi, 7 hours ago, message: Merge pull request #74 from wuharvey/semant-mounika

8ec4c14 was habdulkafi, 7 hours ago, message: fixed deleting of non-poly funs

743bdf9 was kpet, 7 hours ago, message: switching branches changed synth to read channels

e1cc90a was Harvey Wu, 8 hours ago, message: Removed wrong tests

68eb6e7 was Lakshmi Bodapati, 8 hours ago, message: Merge branch 'synth-2' of https://github.com/wuharvey/music-mike into synth-2

0f12965 was Lakshmi Bodapati, 8 hours ago, message: make_midi c

2d1f56a was Harvey Wu, 8 hours ago, message: Inference for blocks

fb03be5 was Mounika, 8 hours ago, message: parens fixes

9a986d8 was Lakshmi Bodapati, 8 hours ago, message: make_midi Mounika changes, testing on cloud

5076048 was Husam Abdul-Kafi, 8 hours ago, message: Merge pull request #73 from wuharvey/master

f5bc384 was Harvey Wu, 9 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike

c1f4047 was Harvey Wu, 9 hours ago, message: FIXED! INFER IS BACK TO NORMAL

1174871 was Harvey Wu, 9 hours ago, message: Debugging printing for Chord

83c5206 was Harvey Wu, 9 hours ago, message: Fixing inference

c266d3d was kpet123, 9 hours ago, message: Merge pull request #71 from wuharvey/junk-branch

f730a08 was habdulkafi, 9 hours ago, message: Merge branch 'semant-mounika' of https://github.com/wuharvey/music-mike into semant-mounika

4c0c348 was Husam Abdul-Kafi, 9 hours ago, message: Merge pull
request #70 from wuharvey/husam-functions
44 9fe34be was kpet, 9 hours ago, message: rests should work now
45 cfd68b9 was Harvey Wu, 9 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
46 212208b was Harvey Wu, 9 hours ago, message: Merge pull request #69 from wuharvey/semant-mounika
47 596ac10 was Harvey Wu, 10 hours ago, message: Merge branch 'semant-mounika' of https://github.com/wuharvey/music-mike into semant-mounika
48 dc907df was Harvey Wu, 10 hours ago, message: Semant testing
49 ada5f38 was Mounika, 10 hours ago, message: project proposal update
50 6a192d9 was habdulkafi, 10 hours ago, message: fixed order of arguments in function decls
51 75b28b4 was Mounika, 10 hours ago, message: Harvey's infer
52 1451ec9 was habdulkafi, 10 hours ago, message: FUNCTIONS HERE WE COME
53 07a857f was Mounika, 10 hours ago, message: semant fixed
54 0712686 was kpet, 10 hours ago, message: sorry on master
55 2573816 was Harvey Wu, 11 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
56 1b1a4b5 was habdulkafi, 11 hours ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
57 e3e17bf was habdulkafi, 11 hours ago, message: fixed up tests
58 037d186 was Mounika, 11 hours ago, message: Merge pull request #68 from wuharvey/semant
59 1235262 was Husam Abdul-Kafi, 23 hours ago, message: Merge pull request #67 from wuharvey/synth-in-codegen
60 30ae318 was habdulkafi, 23 hours ago, message: added new exe file (no idea if its different
61 36e5725 was habdulkafi, 24 hours ago, message: fixed length of rhythm
62 e2a2a0c was habdulkafi, 24 hours ago, message: debugging print statements everywhere
63 f47e2f6 was habdulkafi, 24 hours ago, message: added stuff we were missing
64 226bb6e was habdulkafi, 25 hours ago, message: changed test file
65 2592a3e was habdulkafi, 25 hours ago, message: commented out main. debugging prints
66 eb641c2 was habdulkafi, 25 hours ago, message: added chordlengths
3169529 was habdulkafi, 25 hours ago, message: fixed up testall so it links properly
3f15a4f was habdulkafi, 27 hours ago, message: IT ALMOST WORKS. IT'S NOT LINKING PROPERLY
883c57b was habdulkafi, 28 hours ago, message: trying to fix synth in codegen
febce5f was habdulkafi, 32 hours ago, message: changed around a bunch of types
0ab1fae was habdulkafi, 32 hours ago, message: reversed call list
23d0d51 was habdulkafi, 32 hours ago, message: switched floats to doubles
93e322e was kpet123, 32 hours ago, message: Merge pull request #66 from wuharvey/test-cases
f1860c3 was kpet, 32 hours ago, message: more test cases
322fa4a was kpet, 33 hours ago, message: added new error cases
6b047ec was kpet, 33 hours ago, message: made more fail tests cases
a313859 was habdulkafi, 2 days ago, message: fixed up assignment of rhythm list and codegen for chord list
364dd81 was habdulkafi, 2 days ago, message: fixed up some tests
21f3e5e was habdulkafi, 2 days ago, message: added more expressive error messages for some errors
033c0bb was habdulkafi, 2 days ago, message: added printing for rhythm list
189ef73 was kpet, 2 days ago, message: pushing compiling branch
6b0ba65 was Harvey Wu, 2 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
6015176 was kpet, 2 days ago, message: switching branches (refactored chord list ), need to test *)
9e50a35 was habdulkafi, 2 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
932a181 was habdulkafi, 2 days ago, message: added pretty printing for expr(Fun())
3611c25 was Husam Abdul-Kafi, 2 days ago, message: Merge pull request #65 from wuharvey/semant-mounika
901d757 was Husam Abdul-Kafi, 2 days ago, message: Merge pull request #64 from wuharvey/revert-63-semant-mounika
1052c70 was Husam Abdul-Kafi, 2 days ago, message: Revert "Semant mounika"
Ø1d3abc was Husam Abdul-Kafi, 2 days ago, message: Merge pull request #63 from wuharvey/semant-mounika
683951c was habdulkafi, 2 days ago, message: added printing for functions
be2a32e was Lakshmi Bodapati, 2 days ago, message: @
38a5f0b was Mounika, 2 days ago, message: Merge pull request #62 from wuharvey/LRM-edits
737bad0 was Lakshmi Bodapati, 2 days ago, message: aexpr list vs. aexpr problem
afad984 was Lakshmi Bodapati, 2 days ago, message: compile fixes
85e28ab was Lakshmi Bodapati, 2 days ago, message: all the logic is here?
17fb20f was Harvey Wu, 2 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
da24b60 was Lakshmi Bodapati, 2 days ago, message: semant with husam
352a82b was Lakshmi Bodapati, 2 days ago, message: need to replace Afun dummy with Afun with types using the polycalls. Struggling with syntax for mapping Acall to Afun with correct types
758f93b was kpet, 2 days ago, message: pushing
fa93632 was kpet, 2 days ago, message: pushing branch, inference doesn't work with other stuff
bef71bb was Harvey Wu, 2 days ago, message: Starting semantic checker.
69a145f was kpet, 2 days ago, message: going to switch out files from inference
26839b9 was kpet, 2 days ago, message: mergeMerge branch 'synth-in-codegen' of https://github.com/wuharvey/music-mike into synth-in-codegen
0aadcc0 was Mounika, 3 days ago, message: remove Midi test output
14870da was Mounika, 3 days ago, message: Merge pull request #61 from wuharvey/synth-make
9037ae0 was Mounika, 3 days ago, message: working Makefile that produces synth.o
1f0c93a was Lakshmi Bodapati, 3 days ago, message: make file mimic microC
374bf36 was kpet, 3 days ago, message: nice version of actual operators
b1166e9 was kpet, 3 days ago, message: synth in codegen
fc86caa was kpet, 3 days ago, message: getting weird area for internal map function
23b19d6 was kpet, 3 days ago, message: switching branches
ec236fa was kpet, 3 days ago, message: generalized map function
72bf5bb was Harvey Wu, 3 days ago, message: Updated Final Report, changed style stuff.
196982f was kpet, 3 days ago, message: need to change branches
0bc072e was Harvey Wu, 3 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
31ac72f was kpet, 4 days ago, message: added pointer list structs
0969b09 was kpet, 4 days ago, message: yesMerge branch 'master' of https://github.com/wuharvey/music-mike
d8184b5 was Husam Abdul-Kafi, 4 days ago, message: Merge pull request #58 from wuharvey/midi
047f260 was Harvey Wu, 4 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
2bcb519 was habdulkafi, 4 days ago, message: Merge branch 'midi' of https://github.com/wuharvey/music-mike into midi
da8fb62 was habdulkafi, 4 days ago, message: commented the big print function
e079301 was Mounika, 4 days ago, message: Merge pull request #57 from wuharvey/midi
145e728 was Lakshmi Bodapati, 4 days ago, message: fixing oopsie
6c5ea40 was Lakshmi Bodapati, 4 days ago, message: remove midi
68bd854 was Lakshmi Bodapati, 4 days ago, message: Merge remote-tracking branch 'origin' into midi
dc21dd6 was Lakshmi Bodapati, 4 days ago, message: oops and CFugue deleted
76f8e19 was Lakshmi Bodapati, 4 days ago, message: remove CFugue repos
984d4d7 was Mounika, 4 days ago, message: compiling synth with the executable to produce the Midi thingy
4dce401 was Lakshmi Bodapati, 4 days ago, message: synth slash fix
20b593d was Mounika, 4 days ago, message: song
5a4001d was Mounika, 4 days ago, message: C testing
4df745d was Mounika, 4 days ago, message: testing
751ae46 was Mounika, 4 days ago, message: C fixeseseseseses
64187be was Lakshmi Bodapati, 4 days ago, message: synth to linux

e7e9dab was Harvey Wu, 4 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike

a6af71b was habdulkafi, 4 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike

924378f was habdulkafi, 4 days ago, message: fixed subset?

f6b78ef was Harvey Wu, 4 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike

0de5779 was Harvey Wu, 4 days ago, message: Fixed subset to type typeof return

97072a9 was Lakshmi Bodapati, 4 days ago, message: exec

f1c6b92 was Mounika, 4 days ago, message: Merge pull request #56 from wuharvey/final_report

d1e8865 was Lakshmi Bodapati, 4 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike into midi

ec8969f was Husam Abdul-Kafi, 4 days ago, message: Merge pull request #55 from wuharvey/husam-struct-arrays

791c766 was habdulkafi, 4 days ago, message: added more expressive not found error + took away list.rev

ad3a012 was habdulkafi, 4 days ago, message: fixed list type to be struct

86db086 was habdulkafi, 4 days ago, message: added Printlist function type

0ab3584 was Husam Abdul-Kafi, 4 days ago, message: Merge branch 'master' into husam-struct-arrays

d14caa9 was kpet, 4 days ago, message: getting updated versionMerge branch 'master' of https://github.com/wuharvey/music-mike

db8cc19 was kpet123, 4 days ago, message: Create synth.c

56dad56 was Harvey Wu, 5 days ago, message: Subset type inference

359d996 was Harvey Wu, 5 days ago, message: Useless commit

2f89a3d was Harvey Wu, 5 days ago, message: Debug information for inference only printed with -s flag. Added rhythm list inference

3666fa4 was Mounika, 6 days ago, message: duration syntax

6b9e414 was Mounika, 6 days ago, message: added brackets to numbers

fa0f431 was Mounika, 6 days ago, message: trying out numerical random chords 62q+65q+123q 3a+4h+8h
51fe8da was Mounika, 6 days ago, message: testing a chord
ed4eb40 was Harvey Wu, 6 days ago, message: Codegen takes AEXPR instead of EXPR now.
55d53cc was Harvey Wu, 6 days ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike
7f1e9fd was Harvey Wu, 6 days ago, message: Merge pull request #54 from wuharvey/inference
8d25ebc was Harvey Wu, 6 days ago, message: merge
75595a0 was Harvey Wu, 6 days ago, message: Trying to set up flags for debug messages
e612a0e was Harvey Wu, 6 days ago, message: Merge branch 'inference' of https://github.com/wuharvey/music-mike into inference
2c84db9 was Harvey Wu, 6 days ago, message: Update function types in environment
0f8ae1f was Harvey Wu, 6 days ago, message: Type inference for functions.
c152c94 was habdulkafi, 6 days ago, message: fixed subset into list. added markings for where work is actually being done in iter through list
3633d86 was habdulkafi, 6 days ago, message: GOT LLVM PRINTING OF A LIST TO WORK
b3b4c09 was Harvey Wu, 7 days ago, message: (Almost) no lines wrap over 80 chars now.
e6254cb was Harvey Wu, 7 days ago, message: Merge branch 'master' into inference
8da11fa was Harvey Wu, 7 days ago, message: Added Chord type to ast
d0983f1 was Harvey Wu, 7 days ago, message: Added Keyword Set, Assign inference, style tweaks to infer.ml. Corrected the order of inference. TODO: Replace dummy type in env after Assign.
563f853 was Harvey Wu, 7 days ago, message: Fixed some stuff for strings and pretty printing
8209193 was Harvey Wu, 7 days ago, message: Changed pretty printing to aid debugging. Fixed list inference
611e202 was habdulkafi, 7 days ago, message: fixed up sample app so it takes cmd line args
4869072 was Mounika, 7 days ago, message: sample ap changes that aren't compiling
66abc45 was Mounika, 7 days ago, message: Mounika testing stuff
704eb76 was Harvey Wu, 8 days ago, message: Added parentheses for prec escalation
9286291 was Harvey Wu, 8 days ago, message: Type inference works for If statements
3e2779f was habdulkafi, 8 days ago, message: fixed simple printing error
fb53591 was habdulkafi, 8 days ago, message: added pretty printing for chordlist
93e4360 was Harvey Wu, 8 days ago, message: Added lib.ml for stdlib. Changed Makefile to include lib. Modified musicmike.ml to take -s flag to test type inference. Related changes.
005a03b was habdulkafi, 8 days ago, message: changed structure to only take expressions
04ee773 was habdulkafi, 8 days ago, message: messed with ID wrt Call and Fun
7c9cb0f was habdulkafi, 8 days ago, message: fixed up codegen wrt function definitions
6391ab6 was Harvey Wu, 8 days ago, message: make frontend compiles now
3f32795 was Harvey Wu, 8 days ago, message: Merge pull request #53 from wuharvey/infer
d445d78 was Harvey Wu, 8 days ago, message: Merge branch 'master' into infer
66187ed was Harvey Wu, 8 days ago, message: Added pretty printing for aexprs to AST
0e3f7a4 was Harvey Wu, 9 days ago, message: Changed Makefile and musicmike.ml
682765a was Harvey Wu, 10 days ago, message: Inference module now compiles. Changed fdecl in parser. Minor modifications to ast.ml
be3a01f was Harvey Wu, 2 weeks ago, message: Got rid of user defined types.
192ea8d was Harvey Wu, 2 weeks ago, message: Added more stuff to infer.
3db649f was Lakshmi Bodapati, 2 weeks ago, message: Wrote more stuff for type inference.
ec47966 was Harvey Wu, 2 weeks ago, message: Did a shit ton of stuff yo don't even remember what i did
252748e was Mounika, 2 weeks ago, message: Merge pull request #37 from wuharvey/music-operators
9b25747 was kpet, 2 weeks ago, message: hacked rhythmlist
8ded65a was Lakshmi Bodapati, 2 weeks ago, message: TestPlan
cb5883d was Mounika, 2 weeks ago, message: Merge branch 'master' into music-operators
14c150c was Lakshmi Bodapati, 2 weeks ago, message: architecture section
a198521 was Lakshmi Bodapati, 2 weeks ago, message: I tried to somehow define what each of us worked on but it's kinda hard since we didn't do it by an assigned module basis like other groups seemed to have
a069108 was kpet, 2 weeks ago, message: changed name of pitchlist to chordlist
91b29a7 was kpet, 2 weeks ago, message: added empty rhythm test
8d03848 was kpet, 2 weeks ago, message: compiles- rhythmlist
2b2e880 was kpet, 2 weeks ago, message: added bar to scanner
b41299f was Lakshmi Bodapati, 2 weeks ago, message: rough roles and responsibilities portion
a8c239a was Lakshmi Bodapati, 2 weeks ago, message: Timeline
16fac51 was Lakshmi Bodapati, 2 weeks ago, message: finished project process and style guide
820e911 was Mounika, 2 weeks ago, message: Merge pull request #36 from wuharvey/mounika_semant
0e52b05 was Lakshmi Bodapati, 2 weeks ago, message: merge branch 'master' of https://github.com/wuharvey/music-mike into mounika_semant
6dc6478 was Lakshmi Bodapati, 2 weeks ago, message: Specifications
62f7662 was kpet, 2 weeks ago, message: the last one didn't work- getting weird error : Scanner.is_pat ref -> Lexing.lexbuf -> Parser.token
8742ef2 was kpet, 2 weeks ago, message: fixed indexing error
4b3f962 was kpet, 2 weeks ago, message: it compiles! pitches as expressions
7223852 was kpet, 2 weeks ago, message: getting weird bug in codegen pitches: Error: This expression has type int * 'a but an expression was expected of type 'b * 'c * 'd
9083b6d was Lakshmi Bodapati, 2 weeks ago, message: project plan halfway done
d89d880 was Lakshmi Bodapati, 2 weeks ago, message: added stuff we need from proposal
00e650b was kpet, 2 weeks ago, message: merging with scanner changes Merge branch 'master' of https://github.com/wuharvey/music-mike

35
ac99249 was kpet, 2 weeks ago, message: forgot these
ba681f3 was kpet, 2 weeks ago, message: implemented rlist in codegen, need to sync with semant
3d63c44 was Husam Abdul-Kafi, 2 weeks ago, message: Merge pull request #35 from wuharvey/husam-if-stmnt
76f6c38 was habdulkafi, 2 weeks ago, message: added tests for basic if statements
dd129df was habdulkafi, 2 weeks ago, message: added basic if-then-else functionality
f3dbb5a was kpet, 2 weeks ago, message: llvm seems to be working
6d77a5 was kpet, 2 weeks ago, message: it compiles!
6533bcd was kpet, 2 weeks ago, message: changed orientation again (inner functions sandwiched between mallocs and iters)
3e1b780 was habdulkafi, 2 weeks ago, message: added basic test for function call
24d5cd3 was habdulkafi, 2 weeks ago, message: added basics of function calls
408867d was habdulkafi, 2 weeks ago, message: got rid of reduce/reduce conflicts. added ability to have 0 params fun def and fun calls
3ad1502 was kpet, 2 weeks ago, message: pushing code
fe06d32 was kpet, 2 weeks ago, message: modified version of pitch list no syntax errors at least
787b2e4 was Mounika, 2 weeks ago, message: Merge pull request #33 from wuharvey/mounika_semant
1e3e04c was Mounika, 2 weeks ago, message: change FID
9fc77c3 was Mounika, 2 weeks ago, message: test output
671efd8 was Mounika, 2 weeks ago, message: add rhythm test file
47e260c was Mounika, 2 weeks ago, message: parsing error now!
315da1f was kpet, 2 weeks ago, message: simplified pitch +ast.ml
45418dd was kpet, 2 weeks ago, message: simplitized pitch
a78bcf8 was Mounika, 2 weeks ago, message: compiles
5aab008 was Mounika, 3 weeks ago, message: sorta working...can't quite debug
aab45ba was Lakshmi Bodapati, 3 weeks ago, message: change call to scanner
cfce11d was Lakshmi Bodapati, 3 weeks ago, message: parse
rhythm separately

242 f8d1851 was kpet, 3 weeks ago, message: reformatted list.iteri
243 5d04360 was kpet, 3 weeks ago, message: changing to master
244 4d73d55 was kpet, 3 weeks ago, message: let-in doesn't match, tried reducing problem to list of lists
245 caf80ac was kpet, 3 weeks ago, message: fixed indices
246 9fe5bab was kpet, 3 weeks ago, message: second crack, now at least code makes some sense
247 75e8944 was kpet, 3 weeks ago, message: implemented pitch list still need to test :'
248 41b3ba4 was kpet, 3 weeks ago, message: working on pitchlist
249 9d1edd2 was habdulkafi, 3 weeks ago, message: fixed up pitch list in parser, ast. commented out pitch list in codegen for now
250 9db67f2 was kpet, 3 weeks ago, message: codegen attempt to parser pitch list
251 feffec9 was kpet, 3 weeks ago, message: annotated parser. Issues: cannot have empty lists
252 210e481 was kpet, 3 weeks ago, message: it compiles
git add parser.ml
ast.ml
git add parser.ml
ast.ml
253 604a686 was kpet, 3 weeks ago, message: getting weird error: This expression has type Ast.chord = Ast.pitch list but an expression was expected of type Ast.pitch = int list * int * int list
254 4033493 was kpet, 3 weeks ago, message: working on ast/parser structure
255 dd8124c was kpet, 3 weeks ago, message: scanner and parser with logically consistant updates
256 ccc12d7 was kpet, 3 weeks ago, message: restructured parser so pitches easier to access
257 62afa61 was kpet, 4 weeks ago, message: changed parser and ast
258 3bb8123 was kpet, 4 weeks ago, message: added operations in codegen
259 5580c7b was kpet, 4 weeks ago, message: first commit in pitch operation edits
260 dd6a056 was kpet, 4 weeks ago, message: added LRM and edited some parts
261 f864f7a was kpet, 4 weeks ago, message: nothing worth saving on Kaitlin end Merge branch 'master' of https://github.com/wuharvey/music-mike
262 c5ea544 was kpet, 4 weeks ago, message: saving codegen and
scanner so can pull
263  e28b6bf was Husam Abdul-Kafi, 4 weeks ago, message: Merge pull request #31 from wuharvey/husam-fun-decl
264  a15e49d was habdulkafi, 4 weeks ago, message: added basic function declaration codegen and test file
265  7e7563e was Harvey Wu, 4 weeks ago, message: Removed mike-files directory.
266  99be096 was Lakshmi Bodapati, 4 weeks ago, message: scanner change
267  2a534f0 was Lakshmi Bodapati, 4 weeks ago, message: testing syntax
268  4fdc20f3 was Lakshmi Bodapati, 4 weeks ago, message: more tests
269  e91f565 was Lakshmi Bodapati, 4 weeks ago, message: test
270  97e9950 was Lakshmi Bodapati, 4 weeks ago, message: testing
271  4f6c493 was Lakshmi Bodapati, 4 weeks ago, message: oopsie
272  afe966 was Lakshmi Bodapati, 4 weeks ago, message: testing
273  3316a04 was Lakshmi Bodapati, 4 weeks ago, message: scanner
274  426d243 was Lakshmi Bodapati, 4 weeks ago, message: new approach
275  ff3495f was Lakshmi Bodapati, 4 weeks ago, message: add seed for rhythm list
276  86d53dd was Lakshmi Bodapati, 4 weeks ago, message: parser compiles with no conflicts
277  15b8b06 was Lakshmi Bodapati, 4 weeks ago, message: Rlist
278  254bda2 was Lakshmi Bodapati, 4 weeks ago, message: changed parser and scanner to scan in rhythm list
279  58149cc was Husam Abdul-Kafi, 4 weeks ago, message: Merge pull request #30 from wuharvey/husam-fun-def
280  8e59d52 was habdulkafi, 4 weeks ago, message: added test to test block return
281  ddafd82 was habdulkafi, 4 weeks ago, message: reversed list of exprs in block
282  da614ff was habdulkafi, 4 weeks ago, message: added code generation for blocks
283  8ab5162 was Husam Abdul-Kafi, 4 weeks ago, message: Merge pull request #29 from wuharvey/husam-fun-def
284  bf36fe8 was habdulkafi, 4 weeks ago, message: fixing differences in the merge
285  b7aebef was habdulkafi, 4 weeks ago, message: added parens in fn calls. fixed test files for prints
286  beb578e was kpet123, 4 weeks ago, message: Update parser.mly
287  366d161 was kpet123, 4 weeks ago, message: Merge pull request
#28 from wuharvey/temp
288 a63dc12 was kpet, 4 weeks ago, message: deleted test files
289 99220ce was Harvey Wu, 4 weeks ago, message: Merge branch 'mounika_semant' of https://github.com/wuharvey/music-mike into mounika_semant
290 ec5900a was Harvey Wu, 4 weeks ago, message: removed EOF
291 0de0e08 was kpet, 4 weeks ago, message: added test files
292 4cdc308 was Harvey Wu, 4 weeks ago, message: Corrected spelling for RList
293 d89408f was Lakshmi Bodapati, 4 weeks ago, message: make uppercase in ast
294 a4d8a31 was Lakshmi Bodapati, 4 weeks ago, message: pretty print rhythm list
295 ad48a09 was Lakshmi Bodapati, 4 weeks ago, message: ast to include rhythm list
296 f45108c was Lakshmi Bodapati, 4 weeks ago, message: mounika list attempts
297 3c8db43 was Lakshmi Bodapati, 4 weeks ago, message: possible working scanner lists?
298 bf2a686 was habdulkafi, 4 weeks ago, message: added ; in parser and beginning of function declarations in codegen. BAD STATE
299 c269baa was habdulkafi, 4 weeks ago, message: changed test files to have ;
300 cee5ee9 was Lakshmi Bodapati, 4 weeks ago, message: merge Merge branch 'master' of https://github.com/wuharvey/music-mike into mounika_semant
301 d18378f was Lakshmi Bodapati, 4 weeks ago, message: fix all function calls
302 230d8ca was Harvey Wu, 4 weeks ago, message: Updated string stuff for scanner
303 e561352 was Lakshmi Bodapati, 4 weeks ago, message: L and R paren stuff + fixed a bad merge
304 f332007 was Lakshmi Bodapati, 4 weeks ago, message: merged
305 32531a9 was Lakshmi Bodapati, 4 weeks ago, message: parenthesis to functions compiles with I just make the parser parenthesis to parsing function calls
306 647e542 was Lakshmi Bodapati, 4 weeks ago, message: add parenthesis to parsing function calls
307 22b3205 was Lakshmi Bodapati, 4 weeks ago, message: commenting out stuff I don't think we need
308 3341ebd was habdulkafi, 4 weeks ago, message: fixed up error files for tests. modified codegen to output error
f10564c was habdulkafi, 5 weeks ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike i'm not sure what's going on..
4f3c550 was habdulkafi, 5 weeks ago, message: fixed issue #26 ! added a list.rev to exprs list. fixed test cases
d8fd50e was Husam Abdul-Kafi, 5 weeks ago, message: Merge pull request #27 from wuharvey/husam-arrays
1800b54 was habdulkafi, 5 weeks ago, message: added subsetting to codegen and test files
54314d8 was habdulkafi, 5 weeks ago, message: elements in the list were reversed - fixed now. fixed subsetting
1c2759c was habdulkafi, 5 weeks ago, message: added .[ to scanner
7a2ca69 was habdulkafi, 5 weeks ago, message: changed Sub --> Subset
07ad1e3 was habdulkafi, 5 weeks ago, message: assigning arrays to variables works now
550072e was habdulkafi, 5 weeks ago, message: fixed up assignment and lookup using a hash table
bde3c88 was habdulkafi, 5 weeks ago, message: added SEMI after assignment. The output of the parser is still wrong - the expressions are reverse order
26bb9f2 was habdulkafi, 5 weeks ago, message: added tests that *shouldn't fail (but they do)
e74a42a was habdulkafi, 5 weeks ago, message: added verbose output to Makefile
4e3e4d7 was habdulkafi, 5 weeks ago, message: added start of array implementation
d99dd56 was habdulkafi, 6 weeks ago, message: fixed up test files, merged codegen
e821cc8 was habdulkafi, 6 weeks ago, message: fixed float printing. added assignment
bc5e576 was Harvey Wu, 6 weeks ago, message: Merge pull request #25 from wuharvey/Harvey-Codegen2
0469af1 was Husam Abdul-Kafi, 6 weeks ago, message: Merge pull request #24 from wuharvey/Harvey-Codegen1
7aed9be was habdulkafi, 6 weeks ago, message: fixed up codegen and added test set-up
8da1996 was Harvey Wu, 6 weeks ago, message: Added float printing to Codegen. Added test files.
d648034 was Harvey Wu, 6 weeks ago, message: Merge branch 'Harvey-Codegen1' of https://github.com/wuharvey/music-mike
into Harvey-Codegen1
423ce70 was habdulkafi, 6 weeks ago, message: HELLO WORLD WORKS!
330
fbaaab2 was Harvey Wu, 6 weeks ago, message: Merge branch 'Harvey-Codegen1' of https://github.com/wuharvey/music-mike into Harvey-Codegen1
331
2dec649 was habdulkafi, 6 weeks ago, message: p-->P simple makefile edit
332
d01e532 was Harvey Wu, 6 weeks ago, message: Merge branch 'Harvey-Codegen1' of https://github.com/wuharvey/music-mike into Harvey-Codegen1
333
4218f5d was Harvey Wu, 6 weeks ago, message: Changed to capital P
334
4ea71f2 was Mounika, 6 weeks ago, message: fix syntax for far away test file
335
ac8bd84 was Mounika, 6 weeks ago, message: test file syntax fixes
336
b040d13 was Harvey Wu, 6 weeks ago, message: Renamed microc → musicmike in Makefile and .ml file. Added more stuff to codegen while maintaining compilability
337
83b0bd5 was Harvey Wu, 6 weeks ago, message: Update README.md
338
1bf659d was Harvey Wu, 6 weeks ago, message: Merge pull request #21 from wuharvey/Harvey-Codegen1
339
fa036e9 was habdulkafi, 6 weeks ago, message: added string conversion to ast file
340
30f25d5 was habdulkafi, 6 weeks ago, message: codegen is all commented out, but general structure of what we want is still there
341
6d9d337 was habdulkafi, 6 weeks ago, message: commented out unused tokens cause it complained about them being unused
342
08e6e29 was habdulkafi, 6 weeks ago, message: added ocamlbuild workaround so it compiles on my system
343
9ea5b9a was habdulkafi, 6 weeks ago, message: added the microc.ml file (unchanged). added semant.ml file w/ all of it commented out and it returns unit()
344
d9e92e0 was Harvey Wu, 6 weeks ago, message: Removed irrelevant things in codegen.
345
03e16b2 was Harvey Wu, 6 weeks ago, message: Added more pattern matching to type expr in AST.
346
68cf1e4 was Harvey Wu, 6 weeks ago, message: Added auxiliary functions for 3-tuple in header. Changed high-level structure
of grammar to lists of expressions, fdecls, and tdecls. Added
assign_list to specify lists of assignments for tdecl.

63fd50e was Harvey Wu, 6 weeks ago, message: Added make
command for parser/ast combination compilation

94e0249 was Harvey Wu, 7 weeks ago, message: Removed statement
from ast and added relevant types to EXPR. Fixed small issues
in parser.

6bf70cc was Harvey Wu, 7 weeks ago, message: Added Makefile
(only use currently is make clean)

60c03ac was Harvey Wu, 7 weeks ago, message: Merged codegen
effects with master

132db13 was Harvey Wu, 7 weeks ago, message: Cleaned up
Codegen, minor modifications to parser/scanner

032ee30 was Lakshmi Bodapati, 7 weeks ago, message: repo
organization

77266ae was Kaitlin Pet, 7 weeks ago, message: adding tests
to master

b7ee13e was Harvey Wu, 7 weeks ago, message: Merge pull
request #19 from wuharvey/Kaitlin_edits_to_parser

bec83ba was Harvey Wu, 7 weeks ago, message: Merge branch
'master' into Kaitlin_edits_to_parser

7144505 was Harvey Wu, 7 weeks ago, message: Added FID to AST
and added float stuff to codegen

fa776de was Harvey Wu, 7 weeks ago, message: Merge pull
request #18 from wuharvey/cleanup

1f08473 was Harvey Wu, 7 weeks ago, message: Merge branch
'master' into cleanup

dcd2a25 was kpet123, 7 weeks ago, message: ADDED SET

69f015d was kpet123, 7 weeks ago, message: added set

1fb4aa7 was kpet123, 7 weeks ago, message: Update parser.mly

16d40fb was kpet123, 7 weeks ago, message: Update parser.mly

8e19bf5 was Kaitlin Pet, 7 weeks ago, message: about to push
it

2af2b9a was Kaitlin Pet, 7 weeks ago, message: created
enclosed_expression, which can go to id or stuff in parenthense

720079c was Kaitlin Pet, 7 weeks ago, message: moved assign to
primaries and sub to expr

b6e8d1c was Kaitlin Pet, 7 weeks ago, message: made semi semi
for sequencing and connected funct_list explicitly to function
declaration, also created actuals_list

5f9615c was Harvey Wu, 7 weeks ago, message: Fixed function
call SR conflicts by adding a %prec
b52bbc5 was Harvey Wu, 7 weeks ago, message: Negating floats should not require a -. so deleted corresponding rule in parser.

8399c5b was Harvey Wu, 7 weeks ago, message: Added precedence for rhythm/. Fixed other S/R conflicts - five left and an R/R.

4bf34ba was Harvey Wu, 7 weeks ago, message: added a primaries category per python and renamed Fliteral to FloatLit for consistency

9b446dc was Harvey Wu, 7 weeks ago, message: found source of most S/R conflicts and moved to complex expr section

d9d3def was Harvey Wu, 7 weeks ago, message: fixed some yacc errors. now to fix 127 sr conflicts

008ac56 was Harvey Wu, 7 weeks ago, message: Removed references to whitesp_list and rewrote formals/actuals stuff + other minor cleanup

30b8d67 was Husam Abdul-Kafi, 8 weeks ago, message: Ha codegen (#15)

3c01fde was kpet123, 8 weeks ago, message: Update parser.mly

79a043c was Harvey Wu, 8 weeks ago, message: added some defintions of types and fixed style issues for ast.ml

9a8a145 was Harvey Wu, 8 weeks ago, message: Readded formals_list. Fixed style issues. Fixed regexes for scanner.

816d99a was kpet123, 8 weeks ago, message: Update ast.ml

b66a557 was kpet123, 8 weeks ago, message: Update parser.mly

a3f3d07 was kpet123, 8 weeks ago, message: still needs for loop + concat

1780564 was kpet123, 8 weeks ago, message: Update ast.ml

85cd517 was kpet123, 8 weeks ago, message: Update parser.mly

a927f7a was kpet123, 8 weeks ago, message: most changes are commented in code

799cd1c was kpet123, 8 weeks ago, message: Update scanner.mll

c6a1be9 was Harvey Wu, 8 weeks ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike

f4488c1 was Harvey Wu, 8 weeks ago, message: Removed most MicroC stuff from parser. Added proper fdecl.

59b108a was Harvey Wu, 8 weeks ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike

b75cb5f was Harvey Wu, 8 weeks ago, message: Merge branch 'master' of https://github.com/wuharvey/music-mike

5863ba3 was Harvey Wu, 8 weeks ago, message: Added float operations to parser. Added new type FID (function ID) to
scanner for later convenience. Corrected regex for identifiers to account for reserved single characters.

58136e4 was kpet123, 9 weeks ago, message: Update cfg.txt
58136e4 was kpet123, 9 weeks ago, message: Update cfg.txt
4e002d4 was habdulkafi, 10 weeks ago, message: added the tokens to the token list in the parser and the associativity (very wrong, but it’s a start)
5f0b992 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
5f0b992 was Harvey Wu, 8 weeks ago, message: Added float operations to parser. Added new type FID (function ID) to scanner for later convenience. Corrected regex for identifiers to account for reserved single characters.
7be535e was kpet123, 9 weeks ago, message: Update cfg.txt
d44e2bb was kpet123, 9 weeks ago, message: Update cfg.txt
e6648b6 was kpet123, 9 weeks ago, message: Update cfg.txt
1361d6a was kpet123, 9 weeks ago, message: Update cfg.txt
cia233b9 was kpet123, 9 weeks ago, message: Update cfg.txt
21b7115 was kpet123, 9 weeks ago, message: Update cfg.txt
31eb6c8 was kpet123, 9 weeks ago, message: Update cfg.txt
5eb91d5 was kpet123, 9 weeks ago, message: Add files via upload
b611496 was kpet123, 10 weeks ago, message: Update ast.ml
4cbc462 was habdulkafi, 10 weeks ago, message: added the tokens to the token list in the parser and the associativity (very wrong, but it’s a start)
6845db8 was habdulkafi, 10 weeks ago, message: added microc
c0e9527 was habdulkafi, 10 weeks ago, message: added brackets, indexing, fops, pitchops, concat, and float literals to scanner
3d1705a was habdulkafi, 10 weeks ago, message: added microc scanner
0b9047f was Harvey Wu, 3 months ago, message: Update Project Proposal.md
b02d6bf was Harvey Wu, 3 months ago, message: Update Project Proposal.md
752894c was Harvey Wu, 3 months ago, message: Update Project Proposal.md
3ab535a was Harvey Wu, 3 months ago, message: Update Project Proposal.md
01d9e79 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
36d7140 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
a50edc7 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
d72ea47 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
0de79a0 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
7209a75 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
e0727d4 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
ef5b0b6 was Harvey Wu, 3 months ago, message: Update Project Proposal.md
072c5a1 was Harvey Wu, 3 months ago, message: Create README.md
2e89f7 was Harvey Wu, 3 months ago, message: Project Proposal draft
5. Architectural Design

5.1 Block Diagram
Compiler Architecture

- music-mike program
- Scanner
- Parser
- Inference
- LLVM IR
- Codegen
- Semant
- Executables
- Object Files
- Music-mike Standard Library
- CFugue
- External Music Library Linking

Midi Music File

AST

SAST

LLVM IR
5.2 Interfaces

5.2.1 Scanner (Lakshmi, Husam, Harvey)

Relevant Files: scanner.mll

The scanner is written in Ocamlex and takes the .mike input to the compiler and tokenizes it into literals, identifiers, operators and keywords. It removes the white space and block comments. If any character cannot be lexed by the scanner or if any identifier or literal is not syntactically valid, the scanner throws an error. The tokens created by the scanner are used by the Parser to create an Abstract Syntax Tree. The scanner is context sensitive, so uses different pattern matching inside of rhythm and pitch lists.

5.2.2 Parser (Kaitlin, Husam, Harvey, Lakshmi)

Relevant Files: parser.mly

The parser is written in Ocamlyacc and takes in a series of tokens. It uses the grammar described in parser.mly and datatypes defined in ast.ml to generate an Abstract Syntax Tree. In parser.mly, we define the Musicmike context-free grammar using productions. If the tokens produced by the scanner are successfully parsed that means that the .mike file is syntactically (though perhaps not semantically) correct.

Certain operations are also directly translated to values in the parser. For example, the pitch prefield and postfield operators (Raise by 1 octave, Lower
by 1 octave, Raise by half step, and lower by half step) were parsed directly into numerical values to be applied to the note value during the calling of the Synth function.

5.2.3 Type Inference (Harvey)

Relevant Files: infer.ml

The type inference module takes in the untyped AST of expressions and runs Algorithm W (Hindley Milner Type Inference). Type checking is also done in one swift pass - a list, for example, cannot contain elements of two different types. Scope is also handled in this module - a single global map stores variables and their types and thus typing is dynamic. Variables and functions cannot be redefined. Other semantic checking, such as argument counts

5.2.4 Semant (Lakshmi, Husam, Harvey)

Relevant Files: semant.ml

The semant module is an extension of type inference, and handles polymorphic functions. It first finds all the polymorphic functions, then finds all the calls made to these functions and adds in typed function definitions in place of the dummy type polymorphic functions.
5.2.5 The Code Generator (Husam, Kaitlin, Lakshmi)

Relevant Files: codegen.ml

Our Codegen was similar to that of MicroC with the exception of a few key features. First of all, everything is an expression and evaluates to either a value or void. This structure made it easy for us to nest expressions within one another and take advantage of OCaml’s pattern-matching abilities.

Especially notable was our plethora of structures to manipulate lists. To keeps track of lengths of lists, all lists are not pure pointers but consist of a struct where the first field is length and the second field is a pointer to the list itself. Because the second field contains a pointer, it is sufficient for all single-layer lists.

For multi-layer lists such as the Chordlist structure, this struct was repeated across multiple layers. The data structure consists of a chordlist consisting of chords where each chord consists of pitches and each pitch consists of a 3-tuple. In this case, length information was necessary for chordlist and chord, so those structures were the structs as described above. Since length information was not important for pitches (malloced every time at a set length of 3), pointers to pitches were just int *.
6. **Test Plan**

6.1 **Example tests**
minor1 = [11 13 14 16 18 20 22 23 9];
minor2 = [11 13 14 16 18 20 22 23 10];
theme = p:[0 1 3 5 8 7 8 7b 8 6 8 6b 8 5 8 7 8 4# 6 3 2 3 4# 2 1 v7b 5];
r1 = r:[0. .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .2 .2 .2];
r2 = r:[2.5 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .1 .2 .2 .2];
counter = p:[0 5 6 5 4 3 2 4 3 2 1 v7 1 2 v7 1];
r3 = r:[.1 .1 .1 .1 .1 .1 .1 .1 .5 .1 .1 .1 .1 .1 .1 .1 .1];
arb = p:[0 v5 v7 2 4 v7 2 4 6 5 4 6 5 4 3 2 1 v7 v6 v5 3];
r4 = r:[s s s s s s s s s s s s s s s s s s s 1.25];
rone = Synth(theme r1 minor1 50 1);
rtwo = Synth(counter r3 minor1 (50 + 7) 1);
lone = Synth(theme r2 minor1 33 2);
song = Merge(rone lone);
song1 = Merge(song rtwo);
Make_midi(song1 "bach.midi");
duleID = 'MusicMike'

%list_struct = type < { i32, i32* } >
%chordlist_struct = type < { i32, %chord_struct* } >
%chord_struct = type < { i32, i32** } >
%list_struct_f = type < { i32, double* } >

@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@str = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@f1t = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@str.1 = private unnamed_addr constant [3 x i8] c"%c\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%d \00"
@fmt.3 = private unnamed_addr constant [4 x i8] c"%f \00"
@0 = private unnamed_addr constant [10 x i8] c"bach.midi\00"

declare i32 @printf(i8*, ...)

define i32 @main() {
  entry:
  %array_struct = alloca %list_struct
  %length = getelementptr inbounds %list_struct, %list_struct* %array_struct, i32 0, i32 0
  store i32 9, i32* %length
  %array = alloca i32, i32 9
  %elem = getelementptr i32, i32* %array, i32 0
  store i32 11, i32* %elem
  %elem1 = getelementptr i32, i32* %array, i32 1
  store i32 13, i32* %elem1
  %elem2 = getelementptr i32, i32* %array, i32 2
  store i32 14, i32* %elem2
  %elem3 = getelementptr i32, i32* %array, i32 3
  store i32 16, i32* %elem3
  %elem4 = getelementptr i32, i32* %array, i32 4
  store i32 18, i32* %elem4
  %elem5 = getelementptr i32, i32* %array, i32 5
  store i32 20, i32* %elem5
  %elem6 = getelementptr i32, i32* %array, i32 6
  store i32 22, i32* %elem6
  %elem7 = getelementptr i32, i32* %array, i32 7
40  store i32 23, i32* %elem7
41  %elem8 = getelementptr i32, i32* %array, i32 8
42  store i32 9, i32* %elem8
43  %actual_list = getelementptr inbounds %list_struct,
               %list_struct* %array_struct, i32 0, i32 1
44  store i32* %array, i32** %actual_list
45  %minor1 = alloca %list_struct*
46  store %list_struct* %array_struct, %list_struct** %minor1
47  %array_struct9 = alloca %list_struct
48  %length10 = getelementptr inbounds %list_struct,
              %list_struct* %array_struct9, i32 0, i32 0
49  store i32 9, i32* %length10
50  %array11 = alloca i32, i32 9
51  %elem12 = getelementptr i32, i32* %array11, i32 0
52  store i32 11, i32* %elem12
53  %elem13 = getelementptr i32, i32* %array11, i32 1
54  store i32 13, i32* %elem13
55  %elem14 = getelementptr i32, i32* %array11, i32 2
56  store i32 14, i32* %elem14
57  %elem15 = getelementptr i32, i32* %array11, i32 3
58  store i32 16, i32* %elem15
59  %elem16 = getelementptr i32, i32* %array11, i32 4
60  store i32 18, i32* %elem16
61  %elem17 = getelementptr i32, i32* %array11, i32 5
62  store i32 20, i32* %elem17
63  %elem18 = getelementptr i32, i32* %array11, i32 6
64  store i32 22, i32* %elem18
65  %elem19 = getelementptr i32, i32* %array11, i32 7
66  store i32 23, i32* %elem19
67  %elem20 = getelementptr i32, i32* %array11, i32 8
68  store i32 10, i32* %elem20
69  %actual_list21 = getelementptr inbounds %list_struct,
                %list_struct* %array_struct9, i32 0, i32 1
70  store i32* %array11, i32** %actual_list21
71  %minor2 = alloca %list_struct*
72  store %list_struct* %array_struct9, %list_struct** %minor2
73  %malloccall = tail call i8* @malloc(i32 ptrtoint
               (%chordlist_struct* getelementptr (%chordlist_struct,
                %chordlist struct* null, i32 1) to i32))
74  %cl_struct = bitcast i8* %malloccall to %chordlist_struct*
75  %length22 = getelementptr inbounds %chordlist_struct,
               %chordlist_struct* %cl_struct, i32 0, i32 0

54
store i32 27, i32* %length22

%malloccall23 = tail call i8* @malloc(i32 mul (i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32), i32 27))
%chord_pointer_array = bitcast i8* %malloccall23 to %chord_struct*
%pointer_chord_elem_list = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 0

%malloccall24 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct = bitcast i8* %malloccall24 to %chord_struct*
%length25 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct, i32 0, i32 0
store i32 1, i32* %length25

%malloccall26 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch = bitcast i8* %malloccall26 to i32**
%pitch_pointer_elem = getelementptr i32*, i32** %arr_pitch, i32 0

%malloccall27 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array28 = bitcast i8* %malloccall27 to i32*
%prefield_elem = getelementptr i32, i32* %array28, i32 0
store i32 0, i32* %prefield_elem
%scaledegree_r_elem = getelementptr i32, i32* %array28, i32 1
store i32 0, i32* %scaledegree_r_elem
%postfield_elem = getelementptr i32, i32* %array28, i32 2
store i32 0, i32* %postfield_elem
store i32* %array28, i32** %pitch_pointer_elem
%struct_c_pointer = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct, i32 0, i32 1
store i32** %arr_pitch, i32*** %struct_c_pointer
%actual_chord_struct = load %chord_struct, %chord_struct*
%chord_struct
store %chord_struct %actual_chord_struct, %chord_struct*
%pointer_chord_elem_list29 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 1
%malloccall30 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32), i32 27))
null, i32 1) to i32))
102  %chord_struct31 = bitcast i8* %malloccall30 to
    %chord_struct
103  %length32 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct31, i32 0, i32 0
104  store i32 1, i32* %length32
105  %malloccall33 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
106  %arr_pitch34 = bitcast i8* %malloccall33 to i32**
107  %pitch_pointer_elem35 = getelementptr i32*, i32**
    %arr_pitch34, i32 0
108  %malloccall36 = tail call i8* @malloc(i32 mul (i32 ptrtoint
    (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
109  %array37 = bitcast i8* %malloccall36 to i32*
110  %prefield_elem38 = getelementptr i32, i32* %array37, i32 0
111  store i32 0, i32* %prefield_elem38
112  %scaledegreer_elem39 = getelementptr i32, i32* %array37,
    i32 1
113  store i32 1, i32* %scaledegreer_elem39
114  %postfield_elem40 = getelementptr i32, i32* %array37, i32 2
115  store i32 0, i32* %postfield_elem40
116  store i32* %array37, i32** %pitch_pointer_elem35
117  %struct_c_pointer41 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct31, i32 0, i32 1
118  store i32** %arr_pitch34, i32*** %struct_c_pointer41
119  %actual_chord_struct42 = load %chord_struct, %chord_struct*
    %actual_chord_struct42, %chord_struct
120  store %chord_struct %actual_chord_struct42, %chord_struct*%
    pointer_chord_elem_list29
121  %pointer_chord_elem_list43 = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array, i32 2
122  %malloccall44 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct*, %chord_struct*
    null, i32 1) to i32))
123  %chord_struct45 = bitcast i8* %malloccall44 to
    %chord_struct
124  %length46 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct45, i32 0, i32 0
125  store i32 1, i32* %length46
126  %malloccall47 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
127  %arr_pitch48 = bitcast i8* %malloccall47 to i32**
%pitch_pointer_elem49 = getelementptr i32*, i32**
%arr_pitch48, i32 0
%malloccall50 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array51 = bitcast i8* %malloccall50 to i32*
%prefield_elem52 = getelementptr i32, i32* %array51, i32 0
%scaledegreer_elem53 = getelementptr i32, i32* %array51, i32 1
store i32 0, i32* %prefield_elem52
store i32 0, i32* %scaledegreer_elem53
store i32 0, i32* %prefield_elem54
store i32 0, i32* %scaledegreer_elem54
%struct_c_pointer55 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct45, i32 0, i32 1
store i32** %arr_pitch48, i32*** %struct_c_pointer55
%actual_chord_struct56 = load %chord_struct, %chord_struct*
%pointer_chord_elem_list43
%length60 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct59, i32 0, i32 0
store i32 1, i32* %length60
%malloccall58 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct59 = bitcast i8* %malloccall58 to %chord_struct*
%pointer_chord_elem_list57 = getelementptr %chord_struct %chord_struct, %chord_struct* %chord_struct59, i32 3
%malloccall61 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch62 = bitcast i8* %malloccall61 to i32**
%pitch_pointer_elem63 = getelementptr i32*, i32**
%arr_pitch62, i32 0
%malloccall64 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array65 = bitcast i8* %malloccall64 to i32*
%prefield_elem66 = getelementptr i32, i32* %array65, i32 0
store i32 0, i32* %prefield_elem66
%scaledegreer_elem67 = getelementptr i32, i32* %array65, i32 1
store i32 5, i32* %scaledegree_r_elem67
%postfield_elem68 = getelementptr i32, i32* %array65, i32 2
store i32 0, i32* %postfield_elem68
store i32* %array65, i32** %pitch_pointer_elem63
%struct_c_pointer69 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct59, i32 0, i32 1
store i32** %arr_pitch62, i32*** %struct_c_pointer69
%actual_chord_struct70 = load %chord_struct, %chord_struct*
%chord_struct59
store %chord_struct %actual_chord_struct70, %chord_struct* %pointer_chord_elem_list57
%pointer_chord_elem_list71 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 4
%malloccall72 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct73 = bitcast i8* %malloccall72 to %chord_struct*
%length74 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct73, i32 0, i32 0
store i32 1, i32* %length74
%malloccall75 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch76 = bitcast i8* %malloccall75 to i32**
%pitch_pointer_elem77 = getelementptr i32*, i32** %arr_pitch76, i32 0
%malloccall78 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array79 = bitcast i8* %malloccall78 to i32*
%prefield_elem80 = getelementptr i32, i32* %array79, i32 0
store i32 0, i32* %prefield_elem80
%scaledegree_r_elem81 = getelementptr i32, i32* %array79, i32 1
store i32 8, i32* %scaledegree_r_elem81
%postfield_elem82 = getelementptr i32, i32* %array79, i32 2
store i32 0, i32* %postfield_elem82
store i32* %array79, i32** %pitch_pointer_elem77
%struct_c_pointer83 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct73, i32 0, i32 1
store i32** %arr_pitch76, i32*** %struct_c_pointer83
%actual_chord_struct84 = load %chord_struct, %chord_struct*
%chord_struct73
store %chord_struct %actual_chord_struct84, %chord_struct* %pointer_chord_elem_list85

%pointer_chord_elem_list85 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 5

%malloccall86 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))

%chord_struct87 = bitcast i8* %malloccall86 to %chord_struct*

%length88 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct87, i32 0, i32 0

store i32 1, i32* %length88

%malloccall89 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))

%arr_pitch90 = bitcast i8* %malloccall89 to i32**

%pitch_pointer_elem91 = getelementptr i32*, i32** %arr_pitch90, i32 0

%malloccall92 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))

%array93 = bitcast i8* %malloccall92 to i32*

%prefield_elem94 = getelementptr i32, i32* %array93, i32 0

store i32 0, i32* %prefield_elem94

%scaledegreer_elem95 = getelementptr i32, i32* %array93, i32 1

store i32 7, i32* %scaledegreer_elem95

%postfield_elem96 = getelementptr i32, i32* %array93, i32 2

store i32 0, i32* %postfield_elem96

store i32 0, i32* %prefield_elem94

%actual_chord_struct98 = load %chord_struct, %chord_struct* %chord_struct87

store i32 2, i32* %actual_chord_struct98

%pointer_chord_elem_list85

%pointer_chord_elem_list99 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 6

%malloccall100 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))

%chord_struct101 = bitcast i8* %malloccall100 to %chord_struct*
%length102 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct101, i32 0, i32 0
store i32 1, i32* %length102
%malloccall103 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch104 = bitcast i8* %malloccall103 to i32**
%pitch_pointer_elem105 = getelementptr i32*, i32** %arr_pitch104, i32 0
%malloccall106 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array107 = bitcast i8* %malloccall106 to i32*
%prefield_elem108 = getelementptr i32, i32* %array107, i32 0
store i32 0, i32* %prefield_elem108
%scaledegreer_elem109 = getelementptr i32, i32* %array107, i32 1
store i32 8, i32* %scaledegreer_elem109
%postfield_elem110 = getelementptr i32, i32* %array107, i32 2
store i32 0, i32* %postfield_elem110
store i32* %array107, i32** %pitch_pointer_elem105
%struct_c_pointer111 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct101, i32 0, i32 1
store i32** %arr_pitch104, i32*** %struct_c_pointer111
%actual_chord_struct112 = load %chord_struct, %chord_struct* %chord_struct111
%actual_chord_struct112, %chord_struct*
%pointer_chord_elem_list99
%pointer_chord Elem_list113 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 7
%malloccall1114 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct*, i32 null, i32 1) to i32))
%chord_struct115 = bitcast i8* %malloccall1114 to %chord_struct*
%length116 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct115, i32 0, i32 0
store i32 1, i32* %length116
%malloccall1117 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch118 = bitcast i8* %malloccall1117 to i32**
%pitch_pointer_elem119 = getelementptr i32*, i32**
%arr_pitch118, i32 0
%malloccall120 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
%array121 = bitcast i8* %malloccall120 to i32*
%prefield_elem122 = getelementptr i32, i32* %array121, i32 0
    store i32 0, i32* %prefield_elem122
%scaledegreeer_elem123 = getelementptr i32, i32* %array121, i32 1
    store i32 7, i32* %scaledegreeer_elem123
%postfield_elem124 = getelementptr i32, i32* %array121, i32 2
    store i32 -1, i32* %postfield_elem124
%struct_c_pointer125 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct115, i32 0, i32 1
    store i32*, %arr_pitch118, i32** %struct_c_pointer125
%actual_chord_struct126 = load %chord_struct,
%chord_struct* %chord_struct115
    store %chord_struct %actual_chord_struct126, %chord_struct* %pointer_chord_elem_list113
%pointer_chord_elem_list127 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 8
    %malloccall128 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct129 = bitcast i8* %malloccall128 to
%chord_struct*
%length130 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct129, i32 0, i32 0
    store i32 1, i32* %length130
%malloccall131 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch132 = bitcast i8* %malloccall131 to i32**
%pitch_pointer_elem133 = getelementptr i32*, i32**
%arr_pitch132, i32 0
%malloccall134 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
%array135 = bitcast i8* %malloccall134 to i32*
%prefield_elem136 = getelementptr i32, i32* %array135, i32 0
store i32 0, i32* %prefield_elem136
%scaledegreer_elem137 = getelementptr i32, i32* %array135, i32 1
store i32 8, i32* %scaledegreer_elem137
%postfield_elem138 = getelementptr i32, i32* %array135, i32 2
store i32 0, i32* %postfield_elem138
%struct_c_pointer139 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct129, i32 0, i32 1
store i32** %arr_pitch132, i32*** %struct_c_pointer139
%actual_chord_struct140 = load %chord_struct, %chord_struct* %actual_chord_struct140
store %chord_struct %actual_chord_struct140, %chord_struct* %pointer_chord_elem_list127
%pointer_chord_elem_list141 = getelementptr %chord_struct, %chord_struct* %chord_struct129, %chord_pointer_array, i32 0
%malloccall142 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct143 = bitcast i8* %malloccall142 to %chord_struct
%length144 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct143, i32 0, i32 0
store i32 1, i32* %length144
%malloccall145 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch146 = bitcast i8* %malloccall145 to i32**
%pitch_pointer_elem147 = getelementptr i32*, i32** %arr_pitch146, i32 0
%malloccall148 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array149 = bitcast i8* %malloccall148 to i32*
%prefield_elem150 = getelementptr i32, i32* %array149, i32 0
store i32 0, i32* %prefield_elem150
%scaledegreer_elem151 = getelementptr i32, i32* %array149, i32 1
store i32 6, i32* %scaledegreer_elem151
%postfield_elem152 = getelementptr i32, i32* %array149, i32 2
store i32 0, i32* %postfield_elem152
store i32* %array149, i32** %pitch_pointer_elem147
%struct_c_pointer153 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct143, i32 0, i32 1
store i32** %arr_pitch146, i32*** %struct_c_pointer153
%actual_chord_struct154 = load %chord_struct, %chord_struct* %chord_struct143
store %chord_struct %actual_chord_struct154, %chord_struct* %pointer_chord_elem_list141
%pointer_chord_elem_list155 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 10
%malloccall156 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct157 = bitcast i8* %malloccall156 to %chord_struct*
%length158 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct157, i32 0, i32 0
store i32 1, i32* %length158
%malloccall159 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch160 = bitcast i8* %malloccall159 to i32**
%pitch_pointer_elem161 = getelementptr i32*, i32** %arr_pitch160, i32 0
%malloccall162 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array163 = bitcast i8* %malloccall162 to i32*
%piefield_elem164 = getelementptr i32, i32* %array163, i32 0
store i32 0, i32* %piefield_elem164
%scaledegreer_elem165 = getelementptr i32, i32* %array163, i32 1
store i32 8, i32* %scaledegreer_elem165
%postfield_elem166 = getelementptr i32, i32* %array163, i32 2
store i32 0, i32* %postfield_elem166
store i32* %array163, i32** %pitch_pointer_elem161
%struct_c_pointer167 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct157, i32 0, i32 1
307    store i32** %arr_pitch160, i32*** %struct_c_pointer167
308    %actual_chord_struct168 = load %chord_struct,
   %chord_struct* %chord_struct157
309    store %chord_struct %actual_chord_struct168, %chord_struct*
   %pointer_chord_elem_list155
310    %actual_chord_struct168 = getelementptr %chord_struct,
   %chord_struct* %chord_pointer_array, i32 11
311    %malloccall170 = tail call i8* @malloc(i32 ptrtoint
   (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
312    %chord_struct171 = bitcast i8* %malloccall170 to
   %chord_struct*
313    %length172 = getelementptr inbounds %chord_struct,
   %chord_struct* %chord_struct171, i32 0, i32 0
314    store i32 1, i32* %length172
315    %malloccall173 = tail call i8* @malloc(i32 ptrtoint (i1**
   getelementptr (i1*, i1** null, i32 1) to i32))
316    %arr_pitch174 = bitcast i8* %malloccall173 to i32**
317    %pitch_pointer_elem175 = getelementptr i32*, i32**
   %arr_pitch174, i32 0
318    %malloccall176 = tail call i8* @malloc(i32 mul (i32
   ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
   i32 3))
319    %array177 = bitcast i8* %malloccall176 to i32*
320    %prefield_elem178 = getelementptr i32, i32* %array177, i32
   0
321    store i32 0, i32* %prefield_elem178
322    %scaledegreer_elem179 = getelementptr i32, i32* %array177,
   i32 1
323    store i32 6, i32* %scaledegreer_elem179
324    %postfield_elem180 = getelementptr i32, i32* %array177, i32
   2
325    store i32 -1, i32* %postfield_elem180
326    store i32* %array177, i32** %pitch_pointer_elem175
327    %struct_c_pointer181 = getelementptr inbounds
   %chord_struct, %chord_struct* %chord_struct171, i32 0, i32 1
328    store i32** %arr_pitch174, i32** %struct_c_pointer181
329    %actual_chord_struct182 = load %chord_struct,
   %chord_struct* %chord_struct171
330    store %chord_struct %actual_chord_struct182, %chord_struct*
   %pointer_chord_elem_list169
331    %pointer_chord_elem_list183 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array, i32 12

332  %malloccall184 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
333  %chord_struct185 = bitcast i8* %malloccall184 to %chord_struct*
334  %length186 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct185, i32 0, i32 0
335  store i32 1, i32* %length186
336  %malloccall187 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
337  %arr_pitch188 = bitcast i8* %malloccall187 to i32**
338  %pitch_pointer_elem189 = getelementptr i32*, i32** %arr_pitch188, i32 0
339  %malloccall190 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
340  %array191 = bitcast i8* %malloccall190 to i32*
341  %prefield_elem192 = getelementptr i32, i32* %array191, i32 0
342  store i32 0, i32* %prefield_elem192
343  %scaledegree193 = getelementptr i32, i32* %array191, i32 1
344  store i32 8, i32* %scaledegree193
345  %postfield_elem194 = getelementptr i32, i32* %array191, i32 2
346  store i32 0, i32* %postfield_elem194
347  store i32* %array191, i32** %pitch_pointer_elem189
348  %struct_c_pointer195 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct185, i32 0, i32 1
349  store i32** %arr_pitch188, i32*** %struct_c_pointer195
350  %actual_chord_struct196 = load %chord_struct, %chord_struct* %actual_chord_struct196
351  store %chord_struct %actual_chord_struct196, %chord_struct* %pointer_chord_elem_list183
352  %pointer_chord_elem_list197 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 13
353  %malloccall198 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
354  %chord_struct199 = bitcast i8* %malloccall198 to %chord_struct*
%length200 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct199, i32 0, i32 0
store i32 1, i32* %length200
%malloccall201 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch202 = bitcast i8* %malloccall201 to i32**
%pitch_pointer_elem203 = getelementptr i32*, i32** %arr_pitch202, i32 0
%malloccall204 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array205 = bitcast i8* %malloccall204 to i32*
%prefield_elem206 = getelementptr i32, i32* %array205, i32 0
store i32 0, i32* %prefield_elem206
%scaledegreer_elem207 = getelementptr i32, i32* %array205, i32 1
store i32 5, i32* %scaledegreer_elem207
%postfield_elem208 = getelementptr i32, i32* %array205, i32 2
store i32 0, i32* %postfield_elem208
store i32* %array205, i32** %pitch_pointer_elem203
%struct_c_pointer209 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct199, i32 0, i32 1
store i32** %arr_pitch202, i32*** %struct_c_pointer209
%actual_chord_struct210 = load %chord_struct, %chord_struct* %actual_chord_struct210
store %chord_struct %actual_chord_struct210, %chord_struct* %pointer_chord_elem_list197
%pointer_chord_elem_list211 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 14
%malloccall212 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct* getelementptr (%chord_struct* null, i32 1) to i32))
%chord_struct213 = bitcast i8* %malloccall212 to %chord_struct*
%length214 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct213, i32 0, i32 0
store i32 1, i32* %length214
%malloccall215 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch216 = bitcast i8* %malloccall215 to i32**
%pitch_pointer_elem217 = getelementptr i32*, i32** %arr_pitch216, i32 0
%malloccall218 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array219 = bitcast i8* %malloccall218 to i32*
%prefield_elem220 = getelementptr i32, i32* %array219, i32 0
    store i32 0, i32* %prefield_elem220
%scaledegreer_elem221 = getelementptr i32, i32* %array219, i32 1
    store i32 8, i32* %scaledegreer_elem221
%postfield_elem222 = getelementptr i32, i32* %array219, i32 2
    store i32 0, i32* %postfield_elem222
    store i32* %array219, i32** %pitch_pointer_elem217
%struct_c_pointer223 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct213, i32 0, i32 1
    store i32** %arr_pitch216, i32*** %struct_c_pointer223
%actual_chord_struct224 = load %chord_struct, %chord_struct* %actual_chord_struct224
%pointer_chord_elem_list211
%pointer_chord_elem_list225 = getelementptr %chord_struct* %chord_struct, %chord_struct* %pointer_chord_elem_list211
%malloccall226 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct227 = bitcast i8* %malloccall226 to %chord_struct*
%length228 = getelementptr inbounds %chord_struct* %chord_struct, %chord_struct* %chord_struct227, i32 0, i32 0
    store i32 1, i32* %length228
%malloccall229 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch230 = bitcast i8* %malloccall229 to i32**
%pitch_pointer_elem231 = getelementptr i32*, i32** %arr_pitch230, i32 0
%malloccall232 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array233 = bitcast i8* %malloccall232 to i32*
404  %prefield_elem234 = getelementptr i32, i32* %array233, i32
   0
405   store i32 0, i32* %prefield_elem234
406  %scaledegreer_elem235 = getelementptr i32, i32* %array233,
   i32 1
407   store i32 7, i32* %scaledegreer_elem235
408  %postfield_elem236 = getelementptr i32, i32* %array233, i32
   2
409   store i32 0, i32* %postfield_elem236
410  store i32* %array233, i32** %pitch_pointer_elem231
411  %struct_c_pointer237 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct227, i32 0, i32 1
412   store i32** %arr_pitch230, i32*** %struct_c_pointer237
413  %actual_chord_struct238 = load %chord_struct, %chord_struct*
   %chord_struct238
414   store %chord_struct %actual_chord_struct238, %chord_struct*
   %pointer_chord_elem_list225
415  %pointer_chord_elem_list239 = getelementptr %chord_struct, %chord_struct* %chord_struct, i32 16
416  %malloccall240 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
417  %chord_struct241 = bitcast i8* %malloccall240 to %chord_struct
418  %length242 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct241, i32 0, i32 0
419   store i32 1, i32* %length242
420  %malloccall243 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
421  %arr_pitch244 = bitcast i8* %malloccall243 to i32**
422  %pitch_pointer_elem245 = getelementptr i32*, i32** %arr_pitch244, i32 0
423  %malloccall246 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
424  %array247 = bitcast i8* %malloccall246 to i32*
425  %prefield_elem248 = getelementptr i32, i32* %array247, i32
   0
426   store i32 0, i32* %prefield_elem248
427  %scaledegreer_elem249 = getelementptr i32, i32* %array247, i32
   1
428   store i32 8, i32* %scaledegreer_elem249
%postfieldElem250 = getelementptr i32, i32* %array247, i32 2
store i32 0, i32* %postfieldElem250
store i32* %array247, i32** %pitchPointerElem245
%structCPointer251 = getelementptr inbounds %chordStruct, %chordStruct* %chordStruct241, i32 0, i32 1
store i32** %arrPitch244, i32*** %structCPointer251
%actualChordStruct252 = load %chordStruct,
%chordStruct* %chordStruct241
store %chordStruct %actualChordStruct252, %chordStruct*
%pointerChordElemList239
%pointerChordElemList253 = getelementptr %chordStruct, %chordStruct* %chordPointerArray, i32 17
%mallocCall254 = tail call i8* @malloc(i32 ptrtoint (%chordStruct* getelementptr (%chordStruct, %chordStruct* null, i32 1) to i32))
%chordStruct255 = bitcast i8* %mallocCall254 to %chordStruct*
%length256 = getelementptr inbounds %chordStruct, %chordStruct* %chordStruct255, i32 0, i32 0
store i32 1, i32* %length256
%mallocCall257 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arrPitch258 = bitcast i8* %mallocCall257 to i32**
%pitchPointerElem259 = getelementptr i32*, i32** %arrPitch258, i32 0
%mallocCall260 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array261 = bitcast i8* %mallocCall260 to i32*
%prefieldElem262 = getelementptr i32, i32* %array261, i32 0
store i32 0, i32* %prefieldElem262
%scaledDegreeElem263 = getelementptr i32, i32* %array261, i32 1
store i32 4, i32* %scaledDegreeElem263
%postfieldElem264 = getelementptr i32, i32* %array261, i32 2
store i32 1, i32* %postfieldElem264
store i32* %array261, i32** %pitchPointerElem259
%structCPointer265 = getelementptr inbounds %chordStruct, %chordStruct* %chordStruct255, i32 0, i32 1
store i32** %arr_pitch258, i32*** %struct_c_pointer265
%actual_chord_struct266 = load %chord_struct,
%chord_struct* %chord_struct255
store %chord_struct %actual_chord_struct266, %chord_struct*
%pointer_chord_elem_list253
%pointer_chord_elem_list267 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array, i32 18
%malloccall268 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct269 = bitcast i8* %malloccall268 to
%chord_struct*
%length270 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct269, i32 0, i32 0
store i32 1, i32* %length270
%malloccall271 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch272 = bitcast i8* %malloccall271 to i32**
%pitch_pointer_elem273 = getelementptr i32*, i32**
%arr_pitch272, i32 0
%malloccall274 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
%array275 = bitcast i8* %malloccall274 to i32*
%prefield_elem276 = getelementptr i32, i32* %array275, i32 0
store i32 0, i32* %prefield_elem276
%scaledegreeer_elem277 = getelementptr i32, i32* %array275, i32 1
store i32 6, i32* %scaledegreeer_elem277
%postfield_elem278 = getelementptr i32, i32* %array275, i32 2
store i32 0, i32* %postfield_elem278
store i32* %array275, i32** %pitch_pointer_elem273
%struct_c_pointer279 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct269, i32 0, i32 1
store i32** %arr_pitch272, i32** %struct_c_pointer279
%actual_chord_struct280 = load %chord_struct,
%chord_struct* %chord_struct269
store %chord_struct %actual_chord_struct280, %chord_struct*
%pointer_chord_elem_list267
%pointer_chord_elem_list281 = getelementptr %chord_struct,
502  %length298 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct297, i32 0, i32 0
503    store i32 1, i32* %length298
504  %malloccall299 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
505  %arr_pitch300 = bitcast i8* %malloccall299 to i32**
506  %pitch_pointer_elem301 = getelementptr i32*, i32** %arr_pitch300
507  %pitch_pointer_elem301, i32 0
508  %malloccall302 = tail call i8* @malloc(i32 mul (i32
    ptrtoint (i32* getelementptr (i32, i32** null, i32 1) to i32),
    i32 3))
509  %array303 = bitcast i8* %malloccall302 to i32*
510  %prefield_elem304 = getelementptr i32, i32* %array303,
    i32 0
511  store i32 0, i32* %prefield_elem304
512  %scaledegreeer_elem305 = getelementptr i32, i32* %array303,
    i32 1
513  store i32 2, i32* %scaledegreeer_elem305
514  %postfield_elem306 = getelementptr i32, i32* %array303, i32
    2
515  store i32 0, i32* %postfield_elem306
516  store i32* %array303, i32** %pitch_pointer Elem301
517  %struct_c_pointer307 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct297, i32 0, i32 1
518  store i32** %arr_pitch300, i32*** %struct_c_pointer307
519  %actual_chord_struct308 = load %chord_struct,
    %chord_struct* %chord_struct297
520  store %chord_struct %actual_chord_struct308, %chord_struct* %pointer_chr
521  %pointer_chord_elem list295
522    %pointer_chord_elem_list309 = getelementptr %chord_struct, %chord_struct*
    %chord_pointer_array, i32 21
523  %malloccall310 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct*,
    null, i32 1) to i32))
524  %chord_struct311 = bitcast i8* %malloccall310 to %chord_struct*
525  %length312 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct311, i32 0, i32 0
526  store i32 1, i32* %length312
527  %malloccall313 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
528  %arr_pitch314 = bitcast i8* %malloccall313 to i32**
%pitch_pointer_elem315 = getelementptr i32*, i32*
%arr_pitch314, i32 0
%malloccall316 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array317 = bitcast i8* %malloccall316 to i32*
%prefield_elem318 = getelementptr i32, i32* %array317, i32 0
store i32 0, i32* %prefield_elem318
%scaledegreeer_elem319 = getelementptr i32, i32* %array317, i32 1
store i32 3, i32* %scaledegreeer_elem319
%postfield_elem320 = getelementptr i32, i32* %array317, i32 2
store i32 0, i32* %postfield_elem320
store i32* %array317, i32** %pitch_pointer_elem315
%struct_c_pointer321 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct311, i32 0, i32 1
store i32** %arr_pitch314, i32*** %struct_c_pointer321
%actual_chord_struct322 = load %chord_struct, %chord_struct* %actual_chord_struct322, %chord_struct* %pointer_chord_elem_list309
%pointer_chord_elem_list323 = getelementptr %chord_struct, %chord_struct* %pointer_chord_array, i32 22
%malloccall324 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct325 = bitcast i8* %malloccall324 to %chord_struct*
%length326 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct325, i32 0, i32 0
store i32 1, i32* %length326
%malloccall327 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch328 = bitcast i8* %malloccall327 to i32**
%pitch_pointer_elem329 = getelementptr i32*, i32** %arr_pitch328, i32 0
%malloccall330 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array331 = bitcast i8* %malloccall330 to i32*
%prefield_elem332 = getelementptr i32, i32* %array331, i32 0
store i32 0, i32* %prefield_elem332
%scaledegreer_elem333 = getelementptr i32, i32* %array331, i32 1
store i32 4, i32* %scaledegreer_elem333
%postfield_elem334 = getelementptr i32, i32* %array331, i32 2
store i32 1, i32* %postfield_elem334
store i32* %array331, i32** %pitch_pointer_elem329
%struct_c_pointer335 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct325, i32 0, i32 1
store i32** %arr_pitch328, i32*** %struct_c_pointer335
%actual_chord_struct336 = load %chord_struct, %chord_struct* %actual_chord_struct336, %chord_struct*
%pointer_chord_elem_list323
%pointer_chord_elem_list337 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 23
%malloccall338 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct339 = bitcast i8* %malloccall338 to %chord_struct*
%length340 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct339, i32 0, i32 0
store i32 1, i32* %length340
%malloccall341 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch342 = bitcast i8* %malloccall341 to i32**
%pitch_pointer_elem343 = getelementptr i32*, i32 0
%arr_pitch342, i32 0
%malloccall344 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array345 = bitcast i8* %malloccall344 to i32*
%prefield_elem346 = getelementptr i32, i32* %array345, i32 0
store i32 0, i32* %prefield_elem346
%scaledegreer_elem347 = getelementptr i32, i32* %array345, i32 1
store i32 2, i32* %scaledegreer_elem347
%postfield_elem348 = getelementptr i32, i32* %array345, i32
 store i32 0, i32* %postfield_elem348

store i32 0, i32* %postfield_elem348
store i32* %array345, i32** %pitch_pointer_elem343

%struct_c_pointer349 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct339, i32 0, i32 1

store i32* %arr_pitch342, i32*** %struct_c_pointer349

%actual_chord_struct350 = load %chord_struct,
%chord_struct* %chord_struct339

store %chord_struct %actual_chord_struct350, %chord_struct*
%pointer_chord_elem_list337

%pointer_chord_elem_list351 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array, i32 24

%malloccall352 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))

%chord_struct353 = bitcast i8* %malloccall352 to
%chord_struct*

%length354 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct353, i32 0, i32 0

store i32 1, i32* %length354
%malloccall355 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch356 = bitcast i8* %malloccall355 to i32**

%pitch_pointer_elem357 = getelementptr i32*, i32**
%arr_pitch356, i32 0

%malloccall358 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
%array359 = bitcast i8* %malloccall358 to i32*
%prefield_elem360 = getelementptr i32, i32* %array359, i32
0
store i32 0, i32* %prefield_elem360
%scaledegreer_elem361 = getelementptr i32, i32* %array359,
%array359, i32 1
store i32 1, i32* %scaledegreer_elem361
%postfield_elem362 = getelementptr i32, i32* %array359, i32
2
store i32 0, i32* %postfield_elem362
store i32* %array359, i32** %pitch_pointer_elem357

%struct_c_pointer363 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct353, i32 0, i32 1
store i32** %arr_pitch356, i32*** %struct_c_pointer363
%actual_chord_struct364 = load %chord_struct, %chord_struct* %chord_struct353
store %chord_struct %actual_chord_struct364, %chord_struct*
%pointer_chord_elem_list351
%pointer_chord_elem_list365 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array, i32 25
%malloccall366 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct367 = bitcast i8* %malloccall366 to %chord_struct*
%length368 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct367, i32 0, i32 0
store i32 1, i32* %length368
%malloccall369 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch370 = bitcast i8* %malloccall369 to i32**
%pitch_pointer_elem371 = getelementptr i32*, i32** %arr_pitch370, i32 0
%malloccall372 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array373 = bitcast i8* %malloccall372 to i32*
%prefield_elem374 = getelementptr i32, i32* %array373, i32 0
store i32 -1, i32* %prefield_elem374
%scaledegreer_elem375 = getelementptr i32, i32* %array373, i32 1
store i32 7, i32* %scaledegreer_elem375
%postfield_elem376 = getelementptr i32, i32* %array373, i32 2
store i32 -1, i32* %postfield_elem376
store i32* %array373, i32** %pitch_pointer_elem371
%struct_c_pointer377 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct367, i32 0, i32 1
store i32** %arr_pitch370, i32** %struct_c_pointer377
%actual_chord_struct378 = load %chord_struct, %chord_struct* %chord_struct367
store %chord_struct %actual_chord_struct378, %chord_struct%pointer_chord_elem_list365
%pointer_chord_elem_list379 = getelementptr %chord_struct, %chord_struct* %chord_struct367, i32 0, i32 2
store i32 -1, i32* %postfield_elem376
store i32* %array373, i32** %pitch_pointer_elem371
%struct_c_pointer377 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct367, i32 0, i32 1
store i32** %arr_pitch370, i32** %struct_c_pointer377
%actual_chord_struct378 = load %chord_struct, %chord_struct* %chord_struct367
store %chord_struct %actual_chord_struct378, %chord_struct* %pointer_chord_elem_list365
%pointer_chord_elem_list379 = getelementptr %chord_struct, %chord_struct* %chord_struct367
%chord_struct* %chord_pointer_array, i32 26
626  %malloccall380 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
627  %chord_struct381 = bitcast i8* %malloccall380 to %chord_struct*
628  %length382 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct381, i32 0, i32 0
629  store i32 1, i32* %length382
630  %malloccall383 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
631  %arr_pitch384 = bitcast i8* %malloccall383 to i32**
632  %pitch_pointer_elem385 = getelementptr i32*, i32** %arr_pitch384, i32 0
633  %malloccall386 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
634  %array387 = bitcast i8* %malloccall386 to i32*
635  %prefield_elem388 = getelementptr i32, i32* %array387, i32 0
636  store i32 0, i32* %prefield_elem388
637  %scaledegreeer_elem389 = getelementptr i32, i32* %array387, i32 1
638  store i32 5, i32* %scaledegreeer_elem389
639  %postfield_elem390 = getelementptr i32, i32* %array387, i32 2
640  store i32 0, i32* %postfield_elem390
641  store i32* %array387, i32** %pitch_pointer_elem385
642  %struct_c_pointer391 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct381, i32 0, i32 1
643  store i32** %arr_pitch384, i32*** %struct_c_pointer391
644  %actual_chord struct392 = load %chord struct, %chord struct* %chord struct381
645  store %chord struct %actual_chord struct392, %chord struct* %pointer_chord elem_list379
646  %struct_cl_pointer = getelementptr inbounds %chordlist struct, %chordlist struct* %cl struct, i32 0, i32 1
647  store %chord struct* %chord pointer array, %chord struct* %struct_cl_pointer
648  %theme = alloca %chordlist struct*
649  store %chordlist struct* %cl struct, %chordlist struct**
%array_struct393 = alloca %list_struct_f
%length394 = getelementptr inbounds %list_struct_f, %list_struct_f* %array_struct393, i32 0, i32 0
store i32 27, i32* %length394
%array395 = alloca double, i32 27
%elem396 = getelementptr double, double* %array395, i32 0
store double 0.000000e+00, double* %elem396
%elem397 = getelementptr double, double* %array395, i32 1
store double 1.000000e-01, double* %elem397
%elem398 = getelementptr double, double* %array395, i32 2
store double 1.000000e-01, double* %elem398
%elem399 = getelementptr double, double* %array395, i32 3
store double 1.000000e-01, double* %elem399
%elem400 = getelementptr double, double* %array395, i32 4
store double 1.000000e-01, double* %elem400
%elem401 = getelementptr double, double* %array395, i32 5
store double 1.000000e-01, double* %elem401
%elem402 = getelementptr double, double* %array395, i32 6
store double 1.000000e-01, double* %elem402
%elem403 = getelementptr double, double* %array395, i32 7
store double 1.000000e-01, double* %elem403
%elem404 = getelementptr double, double* %array395, i32 8
store double 1.000000e-01, double* %elem404
%elem405 = getelementptr double, double* %array395, i32 9
store double 1.000000e-01, double* %elem405
%elem406 = getelementptr double, double* %array395, i32 10
store double 1.000000e-01, double* %elem406
%elem407 = getelementptr double, double* %array395, i32 11
store double 1.000000e-01, double* %elem407
%elem408 = getelementptr double, double* %array395, i32 12
store double 1.000000e-01, double* %elem408
%elem409 = getelementptr double, double* %array395, i32 13
store double 1.000000e-01, double* %elem409
%elem410 = getelementptr double, double* %array395, i32 14
store double 1.000000e-01, double* %elem410
%elem411 = getelementptr double, double* %array395, i32 15
store double 1.000000e-01, double* %elem411
%elem412 = getelementptr double, double* %array395, i32 16
store double 1.000000e-01, double* %elem412
%elem413 = getelementptr double, double* %array395, i32 17
store double 1.000000e-01, double* %elem413
%elem414 = getelementptr double, double* %array395, i32 18
store double 1.000000e-01, double* %elem414
%elem415 = getelementptr double, double* %array395, i32 19
store double 1.000000e-01, double* %elem415
%elem416 = getelementptr double, double* %array395, i32 20
store double 1.000000e-01, double* %elem416
%elem417 = getelementptr double, double* %array395, i32 21
store double 1.000000e-01, double* %elem417
%elem418 = getelementptr double, double* %array395, i32 22
store double 1.000000e-01, double* %elem418
%elem419 = getelementptr double, double* %array395, i32 23
store double 1.000000e-01, double* %elem419
%elem420 = getelementptr double, double* %array395, i32 24
store double 1.000000e-01, double* %elem420
%elem421 = getelementptr double, double* %array395, i32 25
store double 2.000000e-01, double* %elem421
%elem422 = getelementptr double, double* %array395, i32 26
store double 2.000000e-01, double* %elem422
%actual_list423 = getelementptr inbounds %list_struct_f, %list_struct_f* %array_struct393, i32 0, i32 1
store double* %array395, double** %actual_list423
%r1 = alloca %list_struct_f*
store %list_struct_f* %array_struct393, %list_struct_f** %r1
%array_struct424 = alloca %list_struct_f
%length425 = getelementptr inbounds %list_struct_f, %list_struct_f* %array_struct424, i32 0, i32 0
store i32 27, i32* %length425
%array426 = alloca double, i32 27
%elem427 = getelementptr double, double* %array426, i32 0
store double 2.500000e+00, double* %elem427
%elem428 = getelementptr double, double* %array426, i32 1
store double 1.000000e-01, double* %elem428
%elem429 = getelementptr double, double* %array426, i32 2
store double 1.000000e-01, double* %elem429
%elem430 = getelementptr double, double* %array426, i32 3
store double 1.000000e-01, double* %elem430
%elem431 = getelementptr double, double* %array426, i32 4
store double 1.000000e-01, double* %elem431
%elem432 = getelementptr double, double* %array426, i32 5
store double 1.000000e-01, double* %elem432
%elem433 = getelementptr double, double* %array426, i32 6
store double 1.000000e-01, double* %elem433
%elem434 = getelementptr double, double* %array426, i32 7
store double 1.000000e-01, double* %elem434
%elem435 = getelementptr double, double* %array426, i32 8
store double 1.000000e-01, double* %elem435
%elem436 = getelementptr double, double* %array426, i32 9
store double 1.000000e-01, double* %elem436
%elem437 = getelementptr double, double* %array426, i32 10
store double 1.000000e-01, double* %elem437
%elem438 = getelementptr double, double* %array426, i32 11
store double 1.000000e-01, double* %elem438
%elem439 = getelementptr double, double* %array426, i32 12
store double 1.000000e-01, double* %elem439
%elem440 = getelementptr double, double* %array426, i32 13
store double 1.000000e-01, double* %elem440
%elem441 = getelementptr double, double* %array426, i32 14
store double 1.000000e-01, double* %elem441
%elem442 = getelementptr double, double* %array426, i32 15
store double 1.000000e-01, double* %elem442
%elem443 = getelementptr double, double* %array426, i32 16
store double 1.000000e-01, double* %elem443
%elem444 = getelementptr double, double* %array426, i32 17
store double 1.000000e-01, double* %elem444
%elem445 = getelementptr double, double* %array426, i32 18
store double 1.000000e-01, double* %elem445
%elem446 = getelementptr double, double* %array426, i32 19
store double 1.000000e-01, double* %elem446
%elem447 = getelementptr double, double* %array426, i32 20
store double 1.000000e-01, double* %elem447
%elem448 = getelementptr double, double* %array426, i32 21
store double 1.000000e-01, double* %elem448
%elem449 = getelementptr double, double* %array426, i32 22
store double 1.000000e-01, double* %elem449
%elem450 = getelementptr double, double* %array426, i32 23
store double 1.000000e-01, double* %elem450
%elem451 = getelementptr double, double* %array426, i32 24
store double 1.000000e-01, double* %elem451
%elem452 = getelementptr double, double* %array426, i32 25
store double 2.000000e-01, double* %elem452
%elem453 = getelementptr double, double* %array426, i32 26
store double 2.000000e-01, double* %elem453
%actual_list454 = getelementptr inbounds %list_struct_f,
%list_struct_f* %array_struct424, i32 0, i32 1
771   store double* %array426, double** %actual_list454
772   %r2 = alloca %list_struct_f*
773   store %list_struct_f* %array_struct424, %list_struct_f** %r2
774   %malloccall455 = tail call i8* @malloc(i32 ptrtoint (%chordlist_struct* getelementptr (%chordlist_struct, %chordlist_struct* null, i32 1) to i32))
775   %cl_struct456 = bitcast i8* %malloccall455 to %chordlist_struct*
776   %length457 = getelementptr inbounds %chordlist_struct, %chordlist_struct* %cl_struct456, i32 0, i32 0
777   store i32 16, i32* %length457
778   %malloccall458 = tail call i8* @malloc(i32 mul (i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32), i32 16))
779   %chord_pointer_array459 = bitcast i8* %malloccall458 to %chord_struct*
780   %pointer_chord_elem_list460 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array459, i32 0
781   %malloccall461 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
782   %chord_struct462 = bitcast i8* %malloccall461 to %chord_struct*
783   %length463 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct462, i32 0, i32 0
784   store i32 1, i32* %length463
785   %malloccall464 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
786   %arr_pitch465 = bitcast i8* %malloccall464 to i32**
787   %pitch_pointer_elem466 = getelementptr i32*, i32** %arr_pitch465, i32 0
788   %malloccall467 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
789   %array468 = bitcast i8* %malloccall467 to i32*
790   %prefield_elem469 = getelementptr i32, i32* %array468, i32 0
791   store i32 0, i32* %prefield_elem469
792   %scaledegreer_elem470 = getelementptr i32, i32* %array468, i32 1
store i32 0, i32* %scaledegree_reelem470
%postfield_elem471 = getelementptr i32, i32* %array468, i32 2
store i32 0, i32* %postfield_elem471
store i32 %array468, i32** %pitch_pointer_elem466
%struct_c_pointer472 = getelementptr inbounds %chord_struct, %chord_struct* %struct_c_pointer462, i32 0, i32 1
store i32* %array468, i32** %pitch_pointer_elem466
%struct_c_pointer472 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct462, i32 0, i32 1
store i32* %arr_pitch465, i32*** %struct_c_pointer472
%actual_chord_struct473 = load %chord_struct, %chord_struct* %actual_chord_struct473, %chord_struct*
%pointer_chord_list460
%pointer_chord_list474 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array459, i32 1
%malloccall475 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct476 = bitcast i8* %malloccall475 to %chord_struct*
%length477 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct476, i32 0, i32 0
store i32 1, i32* %length477
%malloccall478 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch479 = bitcast i8* %malloccall478 to i32**
%pitch_pointer_elem480 = getelementptr i32*, i32** %arr_pitch479, i32 0
%malloccall481 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i1* getelementptr (i13, i13* null, i32 1) to i32), i32 3))
%array482 = bitcast i8* %malloccall481 to i32*
%prefield_elem483 = getelementptr i32, i32* %array482, i32 0
store i32 0, i32* %prefield_elem483
%scaledegree_reelem484 = getelementptr i32, i32* %array482, i32 1
store i32 5, i32* %scaledegree_reelem484
%postfield_elem485 = getelementptr i32, i32* %array482, i32 2
store i32 0, i32* %postfield_elem485
store i32* %array482, i32** %pitch_pointer_elem480
%struct_c_pointer486 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct476, i32 0, i32 1
819    store i32** %arr_pitch479, i32*** %struct_c_pointer486
820    %actual_chord_struct487 = load %chord_struct,
     %chord_struct* %chord_struct476
821    store %chord_struct %actual_chord_struct487, %chord_struct*
     %pointer_chord_elem_list484
822    %pointer_chord_elem_list488 = getelementptr %chord_struct,
     %chord_struct* %chord_pointer_array459, i32 2
823    %alloca489 = tail call i8* @malloc(i32 ptrtoint
     (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
824    %chord_struct490 = bitcast i8* %alloca489 to
     %chord_struct
825    %length491 = getelementptr inbounds %chord_struct,
     %chord_struct* %chord_struct490, i32 0, i32 0
826    store i32 1, i32* %length491
827    %alloca492 = tail call i8* @malloc(i32 ptrtoint (i1**
     getelementptr (i1*, i1** null, i32 1) to i32))
828    %arr_pitch493 = bitcast i8* %alloca492 to i32**
829    %pitch_pointer_elem494 = getelementptr i32*, i32**
     %arr_pitch493, i32 0
830    %alloca495 = tail call i8* @malloc(i32 mul (i32
     ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
     i32 3))
831    %array496 = bitcast i8* %alloca495 to i32*
832    %prefield_elem497 = getelementptr i32, i32* %array496, i32 0
833    store i32 0, i32* %prefield_elem497
834    %scaledegreer_elem498 = getelementptr i32, i32* %array496, i32 1
835    store i32 6, i32* %scaledegreer_elem498
836    %postfield_elem499 = getelementptr i32, i32* %array496, i32 2
837    store i32 0, i32* %postfield_elem499
838    store i32* %array496, i32** %pitch_pointer_elem494
839    %struct_c_pointer500 = getelementptr inbounds
     %chord_struct, %chord_struct* %chord_struct490, i32 0, i32 1
840    store i32** %arr_pitch493, i32*** %struct_c_pointer500
841    %actual_chord_struct501 = load %chord_struct,
     %chord_struct* %chord_struct490
842    store %chord_struct %actual_chord_struct501, %chord_struct*
     %pointer_chord_elem_list488
%pointer_chord_elem_list502 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 3
%malloccall503 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct504 = bitcast i8* %malloccall503 to
%chord_struct*
%length505 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct504, i32 0, i32 0
store i32 1, i32* %length505
%malloccall506 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch507 = bitcast i8* %malloccall506 to i32**
%pitch_pointer_elem508 = getelementptr i32*, i32** %arr_pitch507, i32 0
%malloccall509 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
%array510 = bitcast i8* %malloccall509 to i32*
%priefield_elem511 = getelementptr i32, i32* %array510, i32 0
store i32 0, i32* %priefield_elem511
%scaledegreer_elem512 = getelementptr i32, i32* %array510,
i32 1
store i32 5, i32* %scaledegreer_elem512
%postfield_elem513 = getelementptr i32, i32* %array510, i32 2
store i32 0, i32* %postfield_elem513
store i32 0, i32* %postfield_elem513
store i32* %array510, i32** %pitch_pointer_elem508
%struct_c_pointer514 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct504, i32 0, i32 1
store i32** %arr_pitch507, i32** %struct_c_pointer514
%actual_chord_struct515 = load %chord_struct,
%chord_struct* %chord_struct504
store %chord_struct %actual_chord_struct515, %chord_struct* %pointer_chord_elem_list502
%pointer_chord Elem_list516 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 4
%malloccall517 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct518 = bitcast i8* %malloccall517 to
%chord_struct*
%length519 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct518, i32 0, i32 0
store i32 1, i32* %length519
%malloccall520 = tail call i8* @malloc(i32 ptrtoint (%i1*
getelementptr (%i1*, %i1** null, i32 1) to i32))
%arr_pitch521 = bitcast i8* %malloccall520 to i32**
%pitch_pointer_elem522 = getelementptr i32*, i32**
%arr_pitch521, i32 0
%malloccall523 = tail call i8* @malloc(i32 mul (i32
ptrtoint (%i32* getelementptr (%i32, %i32* null, i32 1) to i32),
i32 3))
%array524 = bitcast i8* %malloccall523 to i32*
%prefield_elem525 = getelementptr i32, i32** %arr_pitch521,
%array524, i32 0
store i32 0, i32* %prefield_elem525
%scaledegreer_elem526 = getelementptr i32, i32** %array524,
i32 1
store i32 4, i32* %scaledegreer_elem526
%postfield_elem527 = getelementptr i32, i32** %array524, i32
2
store i32 0, i32* %postfield_elem527
store i32* %array524, i32** %pitch_pointer_elem522
%struct_c_pointer528 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct518, i32 0, i32 1
store i32* %arr_pitch521, i32*** %struct_c_pointer528
%actual_chord_struct529 = load %chord_struct,
%chord_struct* %chord_struct518
store %chord_struct %actual_chord_struct529, %chord_struct*
%pointer_chord_elem_list516
%pointer_chord_elem_list530 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 5
%malloccall531 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct532 = bitcast i8* %malloccall531 to
%chord_struct*
%length533 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct532, i32 0, i32 0
store i32 1, i32* %length533
%malloccall534 = tail call i8* @malloc(i32 ptrtoint (%i1*
getelementptr (%i1*, %i1** null, i32 1) to i32))
%arr_pitch535 = bitcast i8* %malloccall534 to i32**
%pitch_pointer_elem536 = getelementptr i32*, i32**
%arr_pitch535, i32 0
%malloccall537 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array538 = bitcast i8* %malloccall537 to i32*
%prefield_elem539 = getelementptr i32, i32* %array538, i32 0
store i32 0, i32* %prefield_elem539
%scaledegreeer_elem540 = getelementptr i32, i32* %array538, i32 1
store i32 3, i32* %scaledegreeer_elem540
%postfield_elem541 = getelementptr i32, i32* %array538, i32 2
store i32 0, i32* %postfield_elem541
store i32* %array538, i32** %pitch_pointer_elem536
%struct_c_pointer542 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct532, i32 0, i32 1
store i32** %arr_pitch535, i32*** %struct_c_pointer542
%actual_chord_struct543 = load %chord_struct, %chord_struct* %actual_chord_struct543, %chord_struct* %pointer_chord_elem_list530
store %chord_struct %actual_chord_struct543, %chord_struct* %pointer_chord_elem_list544 = getelementptr %chord_struct %chord_struct, %chord_struct* %chord_pointer_array459, i32 6
%malloccall545 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct546 = bitcast i8* %malloccall545 to %chord_struct*
%length547 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct546, i32 0, i32 0
store i32 1, i32* %length547
%malloccall548 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch549 = bitcast i8* %malloccall548 to i32**
%pitch_pointer_elem550 = getelementptr i32*, i32**
%arr_pitch549, i32 0
%malloccall551 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array552 = bitcast i8* %malloccall551 to i32*
%prefield_elem553 = getelementptr i32, i32* %array552, i32

store i32 0, i32* %prefield_elem553
%scaledegreer_elem554 = getelementptr i32, i32* %array552, i32 1
store i32 2, i32* %scaledegreer_elem554
%postfield_elem555 = getelementptr i32, i32* %array552, i32

store i32 0, i32* %postfield_elem555
store i32* %array552, i32** %pitch_pointer_elem550

%struct_c_pointer556 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct546, i32 0, i32 1
store i32** %arr_pitch549, i32*** %struct_c_pointer556
%actual_chord_struct557 = load %chord_struct, %chord_struct* %actual_chord_struct557, %chord_struct*
%pointer_chord_elem_list544

%malloccall559 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct560 = bitcast i8* %malloccall559 to %chord_struct*

%length561 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct560, i32 0, i32 0
store i32 1, i32* %length561
%malloccall562 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch563 = bitcast i8* %malloccall562 to i32**
%pitch_pointer_elem564 = getelementptr i32*, i32** %arr_pitch563, i32 0
%malloccall565 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array566 = bitcast i8* %malloccall565 to i32*
%prefield_elem567 = getelementptr i32, i32* %array566, i32

store i32 0, i32* %prefield_elem567
%scaledegreer_elem568 = getelementptr i32, i32* %array566, i32 1
store i32 4, i32* %scaledegreeer_elem568
%postfield_elem569 = getelementptr i32, i32* %array566, i32 2
store i32 0, i32* %postfield_elem569
store i32* %pitch_pointer_elem564
%struct_c_pointer570 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct560, i32 0, i32 1
store i32* %arr_pitch563, i32** %struct_c_pointer570
%actual_chord_struct571 = load %chord_struct,
%chord_struct* %chord_struct560
store %chord_struct %actual_chord_struct571, %chord_struct*
%pointer_chord_elem_list558
%pointer_chord Elem_list572 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array459, i32 8
%malloccall573 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct574 = bitcast i8* %malloccall573 to %chord_struct*
%length575 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct574, i32 0, i32 0
store i32 1, i32* %length575
%malloccall576 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch577 = bitcast i8* %malloccall576 to i32**
%pitch_pointer Elem578 = getelementptr i32*, i32** %arr_pitch577, i32 0
%malloccall579 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array580 = bitcast i8* %malloccall579 to i32*
%prefield Elem581 = getelementptr i32, i32* %array580, i32 0
store i32 0, i32* %prefield Elem581
%scaledegreeer Elem582 = getelementptr i32, i32* %array580, i32 1
store i32 3, i32* %scaledegreeer Elem582
%postfield Elem583 = getelementptr i32, i32* %array580, i32 2
store i32 0, i32* %postfield Elem583
store i32* %array580, i32** %pitch_pointer Elem578
%struct_c_pointer584 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct574, i32 0, i32 1
966  store i32** %arr_pitch577, i32*** %struct_c_pointer584
967  %actual_chord_struct585 = load %chord_struct,
%chord_struct* %chord_struct574
968  store %chord_struct %actual_chord_struct585, %chord_struct*
%pointer_chord_elem_list572
969  %pointer_chord_elem_list586 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 9
970  %malloccall587 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
971  %chord_struct588 = bitcast i8* %malloccall587 to %chord_struct*
972  %length589 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct588, i32 0, i32 0
973  store i32 1, i32* %length589
974  %malloccall590 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
975  %arr_pitch591 = bitcast i8* %malloccall590 to i32**
976  %pitch_pointer_elem592 = getelementptr i32*, i32** %arr_pitch591, i32 0
977  %malloccall593 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
978  %array594 = bitcast i8* %malloccall593 to i32*
979  %prefield_elem595 = getelementptr i32, i32* %array594, i32 0
980  store i32 0, i32* %prefield_elem595
981  %scaledegreeer_elem596 = getelementptr i32, i32* %array594, i32 1
982  store i32 2, i32* %scaledegreeer_elem596
983  %postfield_elem597 = getelementptr i32, i32* %array594, i32 2
984  store i32 0, i32* %postfield_elem597
985  store i32* %array594, i32** %pitch_pointer_elem592
986  %struct_c_pointer598 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct588, i32 0, i32 1
987  store i32** %arr_pitch591, i32*** %struct_c_pointer598
988  %actual_chord_struct599 = load %chord_struct,
%chord_struct* %chord_struct588
989  store %chord_struct %actual_chord_struct599, %chord_struct*
%pointer_chord_elem_list586
%pointer_chord_elem_list600 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array459, i32 10
%malloccall601 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct602 = bitcast i8* %malloccall601 to %chord_struct*
%length603 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct602, i32 0, i32 0
store i32 1, i32* %length603
%malloccall604 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch605 = bitcast i8* %malloccall604 to i32**
%pitch_pointer_elem606 = getelementptr i32*, i32** %arr_pitch605, i32 0
%malloccall607 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array608 = bitcast i8* %malloccall607 to i32*
%prefield_elem609 = getelementptr i32, i32* %array608, i32 0
store i32 0, i32* %prefield_elem609
%scaledegreeer_elem610 = getelementptr i32, i32* %array608, i32 1
store i32 1, i32* %scaledegreeer_elem610
%postfield_elem611 = getelementptr i32, i32* %array608, i32 2
store i32 0, i32* %postfield_elem611
store i32* %array608, i32** %pitch_pointer_elem606
%struct_c_pointer612 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct602, i32 0, i32 1
store i32** %arr_pitch605, i32** %struct_c_pointer612
%actual_chord_struct613 = load %chord_struct, %chord_struct* %chord_struct602
store %chord_struct %actual_chord_struct613, %chord_struct* %pointer_chord_elem_list600
%pointer_chord_elem_list614 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array459, i32 11
%malloccall615 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct616 = bitcast i8* %malloccall615 to
%chord_struct*

1014 %length617 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct616, i32 0, i32 0
1015 store i32 1, i32* %length617
1016 %malloccall618 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1017 %arr_pitch619 = bitcast i8* %malloccall618 to i32**
1018 %pitch_pointer_elem620 = getelementptr i32*, i32**
%arr_pitch619, i32 0
1019 %malloccall621 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1020 %array622 = bitcast i8* %malloccall621 to i32*
1021 %prefield_elem623 = getelementptr i32, i32* %array622, i32
0
1022 store i32 -1, i32* %prefield_elem623
1023 %scaledegreeer_elem624 = getelementptr i32, i32* %array622,
 1
1024 store i32 7, i32* %scaledegreeer_elem624
1025 %postfield_elem625 = getelementptr i32, i32* %array622, i32
2
1026 store i32 0, i32* %postfield_elem625
1027 store i32* %array622, i32** %pitch_pointer_elem620
1028 %struct_c_pointer626 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct616, i32 0, i32 1
1029 store i32** %arr_pitch619, i32*** %struct_c_pointer626
1030 %actual_chord_struct627 = load %chord_struct,
%chord_struct* %chord_struct616
1031 store %chord_struct %actual_chord_struct627, %chord_struct*
%pointer_chord_elem_list614
1032 %pointer_chord_elem_list628 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array459, i32 12
1033 %malloccall629 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1034 %chord_struct630 = bitcast i8* %malloccall629 to
%chord_struct*
1035 %length631 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct630, i32 0, i32 0
1036 store i32 1, i32* %length631
1037 %malloccall632 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch633 = bitcast i8* %malloccall632 to i32**
%pitch_pointer_elem634 = getelementptr i32*, i32**
   %arr_pitch633, i32 0
%malloccall635 = tail call i8* @malloc(i32 mul (i32
tp.ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
%array636 = bitcast i8* %malloccall635 to i32*
%prefield_elem637 = getelementptr i32, i32* %array636, i32 0
    store i32 0, i32* %prefield_elem637
%scaledegreeer_elem638 = getelementptr i32, i32* %array636,
   i32 1
    store i32 1, i32* %scaledegreeer_elem638
%postfield_elem639 = getelementptr i32, i32* %array636, i32 2
    store i32 0, i32* %postfield_elem639
store i32* %array636, i32** %pitch_pointer_elem634
%struct_c_pointer640 = getelementptr inbounds
   %chord_struct, %chord_struct* %chord_struct630, i32 0, i32 1
    store i32** %arr_pitch633, i32*** %struct_c_pointer640
%actual_chord_struct641 = load %chord_struct,
   %chord_struct* %chord_struct630
%chord_struct* %actual_chord_struct641, %chord_struct*
   %pointer_chord_elem_list628
%pointer_chord_elem_list642 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array459, i32 13
%malloccall643 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct644 = bitcast i8* %malloccall643 to %chord_struct*
%length645 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct644, i32 0, i32 0
    store i32 1, i32* %length645
%malloccall646 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch647 = bitcast i8* %malloccall646 to i32**
%pitch_pointer_elem648 = getelementptr i32*, i32**
   %arr_pitch647, i32 0
%malloccall649 = tail call i8* @malloc(i32 mul (i32
   ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
%array650 = bitcast i8* %malloccall649 to i32*
%prefield_elem651 = getelementptr i32, i32* %array650, i32 0
  store i32 0, i32* %prefield_elem651
%scaledegreer_elem652 = getelementptr i32, i32* %array650, i32 1
  store i32 2, i32* %scaledegreer_elem652
%postfield_elem653 = getelementptr i32, i32* %array650, i32 2
  store i32 0, i32* %postfield_elem653
  store i32 0, i32* %array650, i32 2 %pitch_pointer_elem648
%struct_c_pointer654 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct644, i32 0, i32 1
  store i32** %arr_pitch647, i32*** %struct_c_pointer654
%actual_chord_struct655 = load %chord_struct, %chord_struct* %actual_chord_struct655
  store %chord_struct %actual_chord_struct655, %chord_struct*
  %pointer_chord_elem_list642 %struct_c_pointer654
%pointer_chord_elem_list656 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array459, i32 14
%malloccall657 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct658 = bitcast i8* %malloccall657 to %chord_struct*
%length659 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct658, i32 0, i32 0
  store i32 1, i32* %length659
%malloccall660 = tail call i8* @malloc(i32 ptrtoint (i1* getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch661 = bitcast i8* %malloccall660 to i32**
%pitch_pointer_elem662 = getelementptr i32*, i32** %arr_pitch661, i32 0
%malloccall663 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array664 = bitcast i8* %malloccall663 to i32*
%prefield_elem665 = getelementptr i32, i32* %array664, i32 0
  store i32 -1, i32* %prefield_elem665
%scaledegreer_elem666 = getelementptr i32, i32* %array664, i32 1
store i32 7, i32* %scaledegreeer_elem666
%postfield_elem667 = getelementptr i32, i32* %array664, i32 2
store i32 0, i32* %postfield_elem667
store i32* %array664, i32** %pitch_pointer_elem662
%struct_c_pointer668 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct658, i32 0, i32 1
store i32** %arr_pitch661, i32*** %struct_c_pointer668
%actual_chord_struct669 = load %chord_struct,
%chord_struct* %chord_struct658
store %chord_struct %actual_chord_struct669, %chord_struct* %pointer_chord_elem_list656
%pointer_chord_elem_list670 = getelementptr %chord_struct, %chord_struct* %pointer_chord_array459, i32 15
%alloca671 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct*, %chord_struct* null, i32 1) to i32))
%chord_struct672 = bitcast i8* %alloca671 to %chord_struct*
%length673 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct672, i32 0, i32 0
store i32 1, i32* %length673
%alloca674 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch675 = bitcast i8* %alloca674 to i32**
%pitch_pointer_elem676 = getelementptr i32*, i32** %arr_pitch675, i32 0
%alloca677 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array678 = bitcast i8* %alloca677 to i32*
%prefield_elem679 = getelementptr i32, i32** %array678, i32 0
store i32 0, i32* %prefield_elem679
%scaledegreeer_elem680 = getelementptr i32, i32* %array678, i32 1
store i32 1, i32* %scaledegreeer_elem680
%postfield_elem681 = getelementptr i32, i32** %pitch_pointer_elem676
%struct_c_pointer682 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct672, i32 0, i32 1
1113 store i32** %arr_pitch675, i32** %struct_c_pointer682
1114 %actual_chord_struct683 = load %chord_struct,
%chord_struct* %chord_struct672
1115 store %chord_struct %actual_chord_struct683, %chord_struct *
%pointer_chord_elem_list670
1116 %struct_cl_pointer684 = getelementptr inbounds
%chordlist_struct, %chordlist_struct* %cl_struct456, i32 0,
i32 1
1117 store %chord_struct* %chord_pointer_array459,
%chord_struct** %struct_cl_pointer684
1118 %counter = alloca %chordlist_struct*
1119 store %chordlist_struct* %cl_struct456, %chordlist_struct**
%counter
1120 %array_struct685 = alloca %list_struct_f
1121 %length686 = getelementptr inbounds %list_struct_f,
%list_struct_f* %array_struct685, i32 0, i32 0
1122 store i32 16, i32* %length686
1123 %array687 = alloca double, i32 16
1124 %elem688 = getelementptr double, double* %array687, i32 0
1125 store double 1.000000e-01, double* %elem688
1126 %elem689 = getelementptr double, double* %array687, i32 1
1127 store double 1.000000e-01, double* %elem689
1128 %elem690 = getelementptr double, double* %array687, i32 2
1129 store double 1.000000e-01, double* %elem690
1130 %elem691 = getelementptr double, double* %array687, i32 3
1131 store double 1.000000e-01, double* %elem691
1132 %elem692 = getelementptr double, double* %array687, i32 4
1133 store double 1.000000e-01, double* %elem692
1134 %elem693 = getelementptr double, double* %array687, i32 5
1135 store double 1.000000e-01, double* %elem693
1136 %elem694 = getelementptr double, double* %array687, i32 6
1137 store double 1.000000e-01, double* %elem694
1138 %elem695 = getelementptr double, double* %array687, i32 7
1139 store double 1.000000e-01, double* %elem695
1140 %elem696 = getelementptr double, double* %array687, i32 8
1141 store double 5.000000e-01, double* %elem696
1142 %elem697 = getelementptr double, double* %array687, i32 9
1143 store double 1.000000e-01, double* %elem697
1144 %elem698 = getelementptr double, double* %array687, i32 10
1145 store double 1.000000e-01, double* %elem698
1146 %elem699 = getelementptr double, double* %array687, i32 11
store double 1.000000e-01, double* %elem699
%elem700 = getelementptr double, double* %array687, i32 12
store double 1.000000e-01, double* %elem700
%elem701 = getelementptr double, double* %array687, i32 13
store double 1.000000e-01, double* %elem701
%elem702 = getelementptr double, double* %array687, i32 14
store double 1.000000e-01, double* %elem702
%elem703 = getelementptr double, double* %array687, i32 15
store double 1.000000e-01, double* %elem703
%actual_list704 = getelementptr inbounds %list_struct_f*, %list_struct_f* %array_struct685, i32 0, i32 1
store double* %array687, double** %actual_list704
%r3 = alloca %list_struct_f*
store %list_struct_f* %array_struct685, %list_struct_f** %r3
%malloccall705 = tail call i8* @malloc(i32 ptrtoint (%chordlist_struct* getelementptr (%chordlist_struct, %chordlist_struct* null, i32 1) to i32))
%cl_struct706 = bitcast i8* %malloccall705 to %chordlist_struct*
%length707 = getelementptr inbounds %chordlist_struct, %chordlist_struct* %cl_struct706, i32 0, i32 0
store i32 21, i32* %length707
%malloccall708 = tail call i8* @malloc(i32 mul (i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32), i32 21))
%chord_pointer_array709 = bitcast i8* %malloccall708 to %chord_struct*
%pointer_chord_elem_list710 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 0
%malloccall711 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct712 = bitcast i8* %malloccall711 to %chord_struct*
%length713 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct712, i32 0, i32 0
store i32 1, i32* %length713
%malloccall714 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch715 = bitcast i8* %malloccall714 to i32**
%pitch_pointer_elem716 = getelementptr i32*, i32**
%arr_pitch715, i32 0
1174  %malloccall717 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
  i32 3))
1175  %array718 = bitcast i8* %malloccall717 to i32*
1176  %prefield_elem719 = getelementptr i32, i32* %array718,
  i32 0
1177  store i32 0, i32* %prefield_elem719
1178  %scaledegreeer_elem720 = getelementptr i32, i32* %array718,
  i32 1
1179  store i32 0, i32* %scaledegreeer_elem720
1180  %postfield_elem721 = getelementptr i32, i32* %array718, i32
  2
1181  store i32 0, i32* %postfield_elem721
1182  store i32* %array718, i32** %pitch_pointer_elem716
1183  %struct_c_pointer722 = getelementptr inbounds %chord_struct,
  %chord_struct* %chord_struct712, i32 0, i32 1
1184  store i32** %arr_pitch715, i32*** %struct_c_pointer722
1185  %actual_chord_struct723 = load %chord_struct,
  %chord_struct* %chord_struct712
1186  store %chord_struct %actual_chord_struct723, %chord_struct*
  %pointer_chord_elem_list710
1187  %pointer_chord_elem_list724 = getelementptr %chord_struct, %chord_struct* %chord_struct712
1188  %malloccall725 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct*
  null, i32 1) to i32))
1189  %chord_struct726 = bitcast i8* %malloccall725 to %chord_struct*
1190  %length727 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct726, i32 0, i32 0
1191  store i32 1, i32* %length727
1192  %malloccall728 = tail call i8* @malloc(i32 ptrtoint (i1**
  getelementptr (i1*, i1** null, i32 1) to i32))
1193  %arr_pitch729 = bitcast i8* %malloccall728 to i32**
1194  %pitch_pointer Elem730 = getelementptr i32*, i32**
  %arr_pitch729, i32 0
1195  %malloccall731 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
  i32 3))
1196  %array732 = bitcast i8* %malloccall731 to i32*
1197  %prefield Elem733 = getelementptr i32, i32* %array732, i32
store i32 -1, i32* %prefield_elem733
%scaledegreer_elem734 = getelementptr i32, i32* %array732, i32 1
store i32 5, i32* %scaledegreer_elem734
%postfield_elem735 = getelementptr i32, i32* %array732, i32 2
store i32 0, i32* %postfield_elem735
store i32* %array732, i32** %pitch_pointer_elem730
%struct_c_pointer736 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct726, i32 0, i32 1
store i32** %arr_pitch729, i32*** %struct_c_pointer736
%actual_chord_struct737 = load %chord_struct, %chord_struct* %actual_chord_struct737, %chord_struct* %pointer_chord_elem_list724
%pointer_chord_elem_list738 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 2
%malloccall739 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct740 = bitcast i8* %malloccall739 to %chord_struct*
%length741 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct740, i32 0, i32 0
store i32 1, i32* %length741
%malloccall742 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch743 = bitcast i8* %malloccall742 to i32**
%pitch_pointer_elem744 = getelementptr i32*, i32** %arr_pitch743, i32 0
%malloccall745 = tail call i8* @malloc(i32 mul (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array746 = bitcast i8* %malloccall745 to i32*
%prefield_elem747 = getelementptr i32, i32* %array746, i32 0
store i32 -1, i32* %prefield_elem747
%scaledegreer_elem748 = getelementptr i32, i32* %array746, i32 1
store i32 7, i32* %scaledegreer_elem748
%postfield_elem749 = getelementptr i32, i32* %array746, i32 2
store i32 0, i32* %postfield_elem749
store i32* %array746, i32** %pitch_pointer_elem744
%struct_c_pointer750 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct740, i32 0, i32 1
store i32** %arr_pitch743, i32*** %struct_c_pointer750
%actual_chord_struct751 = load %chord_struct,
%chord_struct* %chord_struct740
store %chord_struct %actual_chord_struct751, %chord_struct*
%pointer_chord_elem_list738
%pointer_chord_elem_list752 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 3
%malloccall753 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct754 = bitcast i8* %malloccall753 to
%chord_struct*
%length755 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct754, i32 0, i32 0
store i32 1, i32* %length755
%malloccall756 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch757 = bitcast i8* %malloccall756 to i32**
%pitch_pointer_elem758 = getelementptr i32*, %arr_pitch757, i32 0
%malloccall759 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array760 = bitcast i8* %malloccall759 to i32*
%prefield_elem761 = getelementptr i32, %array760, i32
store i32 0, i32* %prefield_elem761
%scaledegreeer_elem762 = getelementptr i32, i32* %array760, i32 1
store i32 2, i32* %scaledegreeer_elem762
%postfield_elem763 = getelementptr i32, i32* %array760, i32
store i32 0, i32* %postfield_elem763
store i32* %array760, i32** %pitch_pointer_elem758
%struct_c_pointer764 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct754, i32 0, i32 1
store i32** %arr_pitch757, i32*** %struct_c_pointer764
1248  %actual_chord_struct765 = load %chord_struct,
    %chord_struct* %chord_struct754
1249  store %chord_struct %actual_chord_struct765, %chord_struct*
    %pointer_chord_elem_list752
1250  %pointer_chord_elem_list766 = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array709, i32 4
1251  %malloccall767 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct*
     null, i32 1) to i32))
1252  %chord_struct768 = bitcast i8* %malloccall767 to
    %chord_struct*
1253  %length769 = getelementptr inbounds %chord_struct,
    %chord_struct* %chord_struct768, i32 0, i32 0
1254  store i32 1, i32* %length769
1255  %malloccall770 = tail call i8* @malloc(i32 ptrtoint (i1**
    getelementptr (i1*, i1** null, i32 1) to i32))
1256  %arr_pitch771 = bitcast i8* %malloccall770 to i32**
1257  %pitch_pointer_elem772 = getelementptr i32*, i32**
    %arr_pitch771, i32 0
1258  %malloccall773 = tail call i8* @malloc(i32 mul (i32
    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
    i32 3))
1259  %array774 = bitcast i8* %malloccall773 to i32*
1260  %prefield_elem775 = getelementptr i32, i32* %array774, i32 0
1261  store i32 0, i32* %prefield_elem775
1262  %scaledegreeer_elem776 = getelementptr i32, i32* %array774,
    i32 1
1263  store i32 4, i32* %scaledegreeer_elem776
1264  %postfield_elem777 = getelementptr i32, i32* %array774, i32 2
1265  store i32 0, i32* %postfield_elem777
1266  store i32* %array774, i32** %pitch_pointer_elem772
1267  %struct_c_pointer778 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct768, i32 0, i32 1
1268  store i32** %arr_pitch771, i32*** %struct_c_pointer778
1269  %actual_chord_struct779 = load %chord_struct,
    %chord_struct* %chord_struct768
1270  store %chord_struct %actual_chord_struct779, %chord_struct*
    %pointer_chord_elem_list766
1271  %pointer_chord_elem_list780 = getelementptr %chord_struct,
    %chord_struct* %chord_pointer_array709, i32 5
1272  %malloccall781 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1273  %chord_struct782 = bitcast i8* %malloccall781 to
%chord_struct*
1274  %length783 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct782, i32 0, i32 0
1275  store i32 1, i32* %length783
1276  %malloccall784 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1277  %arr_pitch785 = bitcast i8* %malloccall784 to i32**
1278  %pitch_pointer_elem786 = getelementptr i32*, i32**
%arr_pitch785, i32 0
1279  %malloccall787 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1280  %array788 = bitcast i8* %malloccall787 to i32*
1281  %prefield_elem789 = getelementptr i32, i32* %array788, i32
0
1282  store i32 -1, i32* %prefield_elem789
1283  %scaledegreeer_elem790 = getelementptr i32, i32* %array788, i32
1
1284  store i32 7, i32* %scaledegreeer_elem790
1285  %postfield_elem791 = getelementptr i32, i32* %array788, i32
2
1286  store i32 0, i32* %postfield_elem791
1287  store i32* %array788, i32** %pitch_pointer_elem786
1288  %struct_c_pointer792 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct782, i32 0, i32 1
1289  store i32** %arr_pitch785, i32*** %struct_c_pointer792
1290  %actual_chord_struct793 = load %chord_struct,
%chord_struct* %chord_struct782
1291  store %chord_struct %actual_chord_struct793, %chord_struct*
%pointer_chord_elem_list780
1292  %pointer_chord_elem_list794 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 6
1293  %malloccall795 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1294  %chord_struct796 = bitcast i8* %malloccall795 to
%chord_struct*
1295  %length797 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct796, i32 0, i32 0
store i32 1, i32* %length797
%malloccall798 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch799 = bitcast i8* %malloccall798 to i32**
%pitch_pointer_elem800 = getelementptr i32*, i32** %arr_pitch799, i32 0
%malloccall801 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i1** getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
%array802 = bitcast i8* %malloccall801 to i32*
%prefield_elem803 = getelementptr i32, i32* %array802, i32 0
store i32 0, i32* %prefield_elem803
%scaledegreeer_elem804 = getelementptr i32, i32* %array802,
i32 1
store i32 2, i32* %scaledegreeer_elem804
%postfield_elem805 = getelementptr i32, i32* %array802, i32 2
store i32 2, i32* %postfield_elem805
store i32* %array802, i32** %pitch_pointer_elem800
%struct_c_pointer806 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct796, i32 0, i32 1
store i32** %arr_pitch799, i32*** %struct_c_pointer806
%actual_chord_struct807 = load %chord_struct,
%chord_struct* %chord_struct796
store %chord_struct %actual_chord_struct807, %chord_struct* %pointer_chord_elem_list794
%pointer_chord_elem_list808 = getelementptr %chord_struct,
%chord_struct* %chord_pointer_array709, i32 7
%malloccall809 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
%chord_struct810 = bitcast i8* %malloccall809 to %chord_struct*
%length811 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct810, i32 0, i32 0
store i32 1, i32* %length811
%malloccall812 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch813 = bitcast i8* %malloccall812 to i32**
%pitch_pointer_elem814 = getelementptr i32*, i32**
%arr_pitch813, i32 0
1321  %malloccall815 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
1322  %array816 = bitcast i8* %malloccall815 to i32*
1323  %prefield_elem817 = getelementptr i32, i32* %array816, i32 0
1324  store i32 0, i32* %prefield_elem817
1325  %scaledegreeer_elem818 = getelementptr i32, i32* %array816, i32 1
1326  store i32 4, i32* %scaledegreeer_elem818
1327  %postfield_elem819 = getelementptr i32, i32* %array816, i32 2
1328  store i32 0, i32* %postfield_elem819
1329  store i32* %array816, i32** %pitch_pointer_elem814
1330  %struct_c_pointer820 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct810, i32 0, i32 1
1331  store i32** %arr_pitch813, i32*** %struct_c_pointer820
1332  %actual_chord_struct821 = load %chord_struct, %chord_struct* %actual_chord_struct821, %chord_struct*
1333  %pointer_chord_elem_list808
1334  %pointer_chord_elem_list822 = getelementptr %chord_struct, %chord_struct* %chord_struct824, i32 8
1335  %malloccall823 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
1336  %chord_struct824 = bitcast i8* %malloccall823 to %chord_struct*
1337  %length825 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct824, i32 0, i32 0
1338  store i32 1, i32* %length825
1339  %malloccall826 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
1340  %arr_pitch827 = bitcast i8* %malloccall826 to i32**
1341  %pitch_pointer_elem828 = getelementptr i32*, i32** %arr_pitch827, i32 0
1342  %malloccall829 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
1343  %array830 = bitcast i8* %malloccall829 to i32*
1344  %prefield_elem831 = getelementptr i32, i32* %array830, i32 0
1345  %actual_chord_struct832 = load %chord_struct, %chord_struct* %actual_chord_struct832, %chord_struct**
1346  %pointer_chord_elem_list822
1347  %pitch_pointer_elem833 = getelementptr i32*, i32** %arr_pitch830, i32 0
1348  %malloccall834 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
1349  %array835 = bitcast i8* %malloccall834 to i32*
1350  %prefield_elem836 = getelementptr i32, i32* %array835, i32 0
1351  %actual_chord_struct837 = load %chord_struct, %chord_struct* %actual_chord_struct837, %chord_struct*
store i32 0, i32* %prefield_elem831
%scaledegreer_elem832 = getelementptr i32, i32* %array830, i32 1
store i32 6, i32* %scaledegreer_elem832
%postfield_elem833 = getelementptr i32, i32* %array830, i32 2
store i32 0, i32* %postfield_elem833
store i32* %array830, i32** %pitch_pointer_elem828
%struct_c_pointer834 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct824, i32 0, i32 1
store i32** %arr_pitch827, i32*** %struct_c_pointer834
%actual_chord_struct835 = load %chord_struct,
%chord_struct* %chord_struct824
store %chord_struct %actual_chord_struct835, %chord_struct*
%pointer_chord_elem_list822
%pointer_chord_elem_list836 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 9
%malloccall837 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct838 = bitcast i8* %malloccall837 to %chord_struct*
%length839 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct838, i32 0, i32 0
store i32 1, i32* %length839
%malloccall840 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch841 = bitcast i8* %malloccall840 to i32**
%pitch_pointer Elem842 = getelementptr i32*, i32** %arr_pitch841, i32 0
%malloccall843 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array844 = bitcast i8* %malloccall843 to i32*
%prefield Elem845 = getelementptr i32, i32* %array844, i32 0
store i32 0, i32* %prefield Elem845
%scaledegreer Elem846 = getelementptr i32, i32* %array844, i32 1
store i32 5, i32* %scaledegreer Elem846
%postfield Elem847 = getelementptr i32, i32* %array844, i32 0
store i32 0, i32* %postfield_elem847
store i32* %array844, i32** %pitch_pointer_elem842
%struct_c_pointer848 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct838, i32 0, i32 1
store i32** %arr_pitch841, i32*** %struct_c_pointer848
%actual_chord_struct849 = load %chord_struct, %chord_struct* %chord_struct838
store %chord_struct %actual_chord_struct849, %chord_struct* %pointer_chord_elem_list836
%pointer_chord_elem_list850 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 10
%malloccall851 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (%chord_struct*, %chord_struct* null, i32 1) to i32))
%chord_struct852 = bitcast i8* %malloccall851 to %chord_struct*
%length853 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct852, i32 0, i32 0
store i32 1, i32* %length853
%malloccall854 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch855 = bitcast i8* %malloccall854 to i32**
%pitch_pointer_elem856 = getelementptr i32*, i32** %arr Pitch855, i32 0
%malloccall857 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array858 = bitcast i8* %malloccall857 to i32*
%prefield_elem859 = getelementptr i32, i32* %array858, i32 0
store i32 0, i32* %prefield_elem859
%scaledegreeer_elem860 = getelementptr i32, i32* %array858, i32 1
store i32 4, i32* %scaledegreeer_elem860
%postfield_elem861 = getelementptr i32, i32* %array858, i32 2
store i32 0, i32* %postfield_elem861
store i32* %array858, i32** %pitch_pointer_elem856
%struct_c_pointer862 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct852, i32 0, i32 1
store i32** %arr_pitch855, i32*** %struct_c_pointer862
%actual_chord_struct863 = load %chord_struct, %chord_struct* %chord_struct852
store %chord_struct %actual_chord_struct863, %chord_struct* %pointer_chord_elem_list850
%pointer_chord_elem_list864 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 11
%malloccall865 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct866 = bitcast i8* %malloccall865 to %chord_struct*
%length867 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct866, i32 0, i32 0
store i32 1, i32* %length867
%malloccall868 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch869 = bitcast i8* %malloccall868 to i32**
%pitch_pointer_elem870 = getelementptr i32*, i32** %arr_pitch869, i32 0
%malloccall871 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array872 = bitcast i8* %malloccall871 to i32*
%prefield_elem873 = getelementptr i32, i32* %array872, i32 0
store i32 0, i32* %prefield_elem873
%scaledegreeer_elem874 = getelementptr i32, i32* %array872, i32 1
store i32 6, i32* %scaledegreeer_elem874
%postfield_elem875 = getelementptr i32, i32* %array872, i32 2
store i32 0, i32* %postfield_elem875
store i32* %array872, i32** %pitch_pointer_elem870
%struct_c_pointer876 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct866, i32 0, i32 1
store i32* %arr_pitch869, i32*** %struct_c_pointer876
%actual_chord_struct877 = load %chord_struct, %chord_struct* %chord_struct866
store %chord_struct %actual_chord_struct877, %chord_struct* %pointer_chord_elem_list864
%pointer_chord_elem_list878 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 12
%malloccall879 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct880 = bitcast i8* %malloccall879 to %chord_struct*
%length881 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct880, i32 0, i32 0
store i32 1, i32* %length881
%malloccall882 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i1* getelementptr (i1*, i1** null, i32 1) to i32), i32 3))
%arr_pitch883 = bitcast i8* %malloccall882 to i32**
%pitch_pointer_elem884 = getelementptr i32*, i32** %arr_pitch883, i32 0
%malloccall885 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%array886 = bitcast i8* %malloccall885 to i32*
%prefield_elem887 = getelementptr i32, i32* %array886, i32 0
store i32 0, i32* %prefield_elem887
%scaledegreer_elem888 = getelementptr i32, i32* %array886, i32 1
store i32 5, i32* %scaledegreer_elem888
%postfield_elem889 = getelementptr i32, i32* %array886, i32 2
store i32 0, i32* %postfield_elem889
store i32* %array886, i32** %pitch_pointer_elem884
%struct_c_pointer890 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct880, i32 0, i32 1
store i32** %arr_pitch883, i32*** %struct_c_pointer890
%actual_chord_struct891 = load %chord_struct, %chord_struct* %chord_struct880
store %chord_struct %actual_chord_struct891, %chord_struct* %pointer_chord_elem_list878
%pointer_chord_elem_list892 = getelementptr %chord_struct, %chord_struct* %chord_struct891, i32 13
%malloccall893 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct894 = bitcast i8* %malloccall893 to %chord_struct*
%length895 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct894, i32 0, i32 0
1443   store i32 1, i32* %length895
1444   %malloccall896 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1445   %arr_pitch897 = bitcast i8* %malloccall896 to i32**
1446   %pitch_pointer_elem898 = getelementptr i32*, i32**
%arr_pitch897, i32 0
1447   %malloccall899 = tail call i8* @malloc(i32 mul (i32
ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
i32 3))
1448   %array900 = bitcast i8* %malloccall899 to i32*
1449   %prefield_elem901 = getelementptr i32, i32* %array900, i32
0
1450   store i32 0, i32* %prefield_elem901
1451   %scaledegreeer_elem902 = getelementptr i32, i32* %array900,
1452   i32 1
1453   store i32 4, i32* %scaledegreeer_elem902
1454   %postfield_elem903 = getelementptr i32, i32* %array900, i32
2
1455   store i32 0, i32* %postfield_elem903
1456   store i32* %array900, i32** %pitch_pointer_elem898
1457   %struct_c_pointer904 = getelementptr inbounds
%chord_struct, %chord_struct* %chord_struct894, i32 0, i32 1
1458   store i32** %arr_pitch897, i32*** %struct_c_pointer904
1459   %actual_chord_struct905 = load %chord_struct,
%chord_struct* %chord_struct894
1460   store %chord_struct %actual_chord_struct905, %chord_struct*
%pointer_chord_elem_list892
1461   %pointer_chord_elem_list906 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 14
1462   %malloccall907 = tail call i8* @malloc(i32 ptrtoint
(%chord_struct* getelementptr (%chord_struct, %chord_struct*
null, i32 1) to i32))
1463   %chord_struct908 = bitcast i8* %malloccall907 to
%chord_struct*
1464   %length909 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct908, i32 0, i32 0
1465   store i32 1, i32* %length909
1466   %malloccall910 = tail call i8* @malloc(i32 ptrtoint (i1**
getelementptr (i1*, i1** null, i32 1) to i32))
1467   %arr_pitch911 = bitcast i8* %malloccall910 to i32**
1468   %pitch_pointer_elem912 = getelementptr i32*, i32**
%arr_pitch911, i32 0
1468 %malloccall913 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
1469 %array914 = bitcast i8* %malloccall913 to i32*
1470 %prefield_elem915 = getelementptr i32, i32* %array914, i32 0
1471 store i32 0, i32* %prefield_elem915
1472 %scaledegreer_elem916 = getelementptr i32, i32* %array914, i32 1
1473 store i32 3, i32* %scaledegreer_elem916
1474 %postfield_elem917 = getelementptr i32, i32* %array914, i32 2
1475 store i32 0, i32* %postfield_elem917
1476 store i32* %array914, i32** %pitch_pointer_elem912
1477 %struct_c_pointer918 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct908, i32 0, i32 1
1478 store i32** %arr_pitch911, i32*** %struct_c_pointer918
1479 %actual_chord_struct919 = load %chord_struct, %chord_struct* %actual_chord_struct919, %chord_struct* %actual_chord_struct_list906
1480 store %chord_struct %actual_chord_struct919, %chord_struct* %pointer_chord_elem_list906
1481 %pointer_chord_elem_list920 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 15
1482 %malloccall921 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
1483 %chord_struct922 = bitcast i8* %malloccall921 to %chord_struct*
1484 %length923 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct922, i32 0, i32 0
1485 store i32 1, i32* %length923
1486 %malloccall924 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
1487 %arr_pitch925 = bitcast i8* %malloccall924 to i32**
1488 %pitch_pointer Elem926 = getelementptr i32*, i32** %arr_pitch925, i32 0
1489 %malloccall927 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
1490 %array928 = bitcast i8* %malloccall927 to i32*
1491 %prefield Elem929 = getelementptr i32, i32* %array928, i32 0
store i32 0, i32* %prefield_elem929
%scaledegreeer_elem930 = getelementptr i32, i32* %array928, i32 1
store i32 2, i32* %scaledegreeer_elem930
%postfield_elem931 = getelementptr i32, i32* %array928, i32 2
store i32 0, i32* %postfield_elem931
store i32* %array928, i32** %pitch_pointer_elem926
%struct_c_pointer932 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct922, i32 0, i32 1
store i32** %arr_pitch925, i32** %struct_c_pointer932
%actual_chord_struct933 = load %chord_struct, %chord_struct* %actual_chord_struct933, %chord_struct* %pointer_chord_elem_list920
%pointer_chord_elem_list934 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 16
%malloccall935 = tail call i8* @malloc(i32 ptrtoint (i1** %arr_pitch925, i32** %struct_c_pointer932 null, i32 1) to i32))
%chord_struct936 = bitcast i8* %malloccall935 to %chord_struct
%length937 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct936, i32 0, i32 0
store i32 1, i32* %length937
%malloccall938 = tail call i8* @malloc(i32 ptrtoint (i1** %arr_pitch925 %malloccall938 to i32))
%pitch_pointer_elem940 = getelementptr i32*, i32** %arr_pitch926
%pitch_pointer_elem941 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* %prefield_elem943 to i32), i32 3))
%array942 = bitcast i8* %malloccall941 to i32
%prefield_elem943 = getelementptr i32, i32* %array942, i32 0
store i32 0, i32* %prefield_elem943
%scaledegreeer_elem944 = getelementptr i32, i32* %array942, i32 1
store i32 1, i32* %scaledegreeer_elem944
%postfield_elem945 = getelementptr i32, i32* %array942, i32 2
store i32 0, i32* %postfield_elem945
store i32* %array942, i32** %pitch_pointer_elem940
%struct_c_pointer946 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct936, i32 0, i32 1
store i32** %arr_pitch939, i32*** %struct_c_pointer946
%actual_chord_struct947 = load %chord_struct,
    %chord_struct* %chord_struct936
store %chord_struct %actual_chord_struct947, %chord_struct*
    %pointer_chord_elem_list934
%pointer_chord_elem_list948 = getelementptr %chord_struct, %chord_struct*> %chord_pointer_array709, i32 17
%malloccall949 = tail call i8* @malloc(i32 ptrtoint
    (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct950 = bitcast i8* %malloccall949 to
    %chord_struct*
%length951 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct950, i32 0, i32 0
store i32 1, i32* %length951
%malloccall952 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch953 = bitcast i8* %malloccall952 to i32**
%pitch_pointer_elem954 = getelementptr i32*, i32** %pitch_pointer_elem953, i32 0
%malloccall955 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr i32, i32 null, i32 1) to i32), i32 3))
%array956 = bitcast i8* %malloccall955 to i32*
%prefield_elem957 = getelementptr i32, i32* %array956, i32 0
store i32 -1, i32* %prefield_elem957
%scaledegreeer_elem958 = getelementptr i32, i32* %array956, i32 1
store i32 7, i32* %scaledegreeer_elem958
%postfield_elem959 = getelementptr i32, i32* %array956, i32 2
store i32 0, i32* %postfield_elem959
store i32* %array956, i32** %pitch_pointer_elem954
%struct_c_pointer960 = getelementptr inbounds
    %chord_struct, %chord_struct* %chord_struct950, i32 0, i32 1
store i32** %arr_pitch953, i32*** %struct_c_pointer960
%actual_chord_struct961 = load %chord_struct, %chord_struct* %chord_struct950
store %chord_struct %actual_chord_struct961, %chord_struct* %pointer_chord_elem_list948
%pointer_chord_elem_list962 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 18
%malloccall963 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
%chord_struct964 = bitcast i8* %malloccall963 to %chord_struct*
%length965 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct964, i32 0, i32 0
store i32 1, i32* %length965
%malloccall966 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
%arr_pitch967 = bitcast i8* %malloccall966 to i32**
%pitch_pointer_elem968 = getelementptr i32*, i32** %arr_pitch967, i32 0
%malloccall969 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
%array970 = bitcast i8* %malloccall969 to i32*
%prefield_elem971 = getelementptr i32, i32* %array970, i32 0
store i32 -1, i32* %prefield_elem971
%scaledegreeer_elem972 = getelementptr i32, i32* %array970, i32 1
store i32 6, i32* %scaledegreeer_elem972
%postfield_elem973 = getelementptr i32, i32* %array970, i32 2
store i32 0, i32* %postfield_elem973
store i32* %array970, i32** %pitch_pointer_elem968
%struct_c_pointer974 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct964, i32 0, i32 1
store i32* %arr_pitch967, i32** %struct_c_pointer974
%actual_chord_struct975 = load %chord_struct, %chord_struct* %chord_struct964
store %chord_struct %actual_chord_struct975, %chord_struct* %pointer_chord_elem_list962
%pointer_chord_elem_list976 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 19
1566 %malloccall977 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
1567 %chord_struct978 = bitcast i8* %malloccall977 to %chord_struct*
1568 %length979 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct978, i32 0, i32 0
1569 store i32 1, i32* %length979
1570 %malloccall980 = tail call i8* @malloc(i32 ptrtoint (i1** getelementptr (i1*, i1** null, i32 1) to i32))
1571 %arr_pitch981 = bitcast i8* %malloccall980 to i32**
1572 %pitch_pointer_elem982 = getelementptr i32*, i32** %arr_pitch981, i32 0
1573 %malloccall983 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32), i32 3))
1574 %array984 = bitcast i8* %malloccall983 to i32*
1575 %prefield_elem985 = getelementptr i32, i32* %array984, i32 0
1576 store i32 -1, i32* %prefield_elem985
1577 %scaledgreer_elem986 = getelementptr i32, i32* %array984, i32 1
1578 store i32 5, i32* %scaledgreer_elem986
1579 %postfield_elem987 = getelementptr i32, i32* %array984, i32 2
1580 store i32 0, i32* %postfield_elem987
1581 store i32* %array984, i32** %pitch_pointer_elem982
1582 %struct_c_pointer988 = getelementptr inbounds %chord_struct, %chord_struct* %chord_struct978, i32 0, i32 1
1583 store i32** %arr_pitch981, i32*** %struct_c_pointer988
1584 %actual_chord_struct989 = load %chord_struct, %chord_struct* %actual_chord_struct989
1585 store %chord_struct %actual_chord_struct989, %chord_struct* %pointer_chord_elem_list976
1586 %pointer_chord_elem_list990 = getelementptr %chord_struct, %chord_struct* %chord_pointer_array709, i32 20
1587 %malloccall991 = tail call i8* @malloc(i32 ptrtoint (%chord_struct* getelementptr (%chord_struct, %chord_struct* null, i32 1) to i32))
1588 %chord_struct992 = bitcast i8* %malloccall991 to %chord_struct*
1589 %length993 = getelementptr inbounds %chord_struct,
%chord_struct* %chord_struct992, i32 0, i32 0
1590    store i32 1, i32* %length993
1591    %malloccall994 = tail call i8* @malloc(i32 ptrtoint (i1**
1592    getelementptr (i1*, i1** null, i32 1) to i32))
1593    %pitch_pointer_elem996 = getelementptr i32*, i32**
1594    %arr_pitch995 = bitcast i8* %malloccall994 to i32**
1595    %malloccall997 = tail call i8* @malloc(i32 mul (i32
1596    ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32),
1597    i32 3))
1598    %array998 = bitcast i8* %malloccall997 to i32*
1599    %prefield_elem999 = getelementptr i32, i32* %array998,
1600    store i32 0, i32* %prefield_elem999
1601    %scaledegreer_elem1000 = getelementptr i32, i32* %array998,
1602    store i32 3, i32* %scaledegreer_elem1000
1603    %postfield_elem1001 = getelementptr i32, i32* %array998,
1604    store i32 0, i32* %postfield_elem1001
1605    %struct_c_pointer1002 = getelementptr inbounds
1606    %chord_struct*, %chord_struct* %chord_struct992, i32 0, i32 1
1607    %actual_chord_struct1003 = load %chord_struct,
1608    %chord_struct* %chord_struct992
1609    store %chord_struct %actual_chord_struct1003,
1610    %chord_struct* %pointer_chord_elem_list990
1611    %struct_cl_pointer1004 = getelementptr inbounds
1612    %chordlist_struct, %chordlist_struct* %cl_struct706, i32 0,
1613    i32 1
1614    store %chord_struct* %chord_pointer_array709,
1615    %chord_struct** %struct_cl_pointer1004
1616    %arp = alloca %chordlist_struct*
1617    store %chordlist_struct* %cl_struct706, %chordlist_struct**
1618    %arp
1619    %array_struct1005 = alloca %list_struct_f
1620    %length1006 = getelementptr inbounds %list_struct_f,
1621    %list_struct_f* %array_struct1005, i32 0, i32 0
1622    store i32 21, i32* %length1006
1623    %array1007 = alloca double, i32 21
1624    %elem1008 = getelementptr double, double* %array1007, i32 0

114
store double 2.500000e-01, double* %elem1008
%elem1009 = getelementptr double, double* %array1007, i32 1
store double 2.500000e-01, double* %elem1009
%elem1010 = getelementptr double, double* %array1007, i32 2
store double 2.500000e-01, double* %elem1010
%elem1011 = getelementptr double, double* %array1007, i32 3
store double 2.500000e-01, double* %elem1011
%elem1012 = getelementptr double, double* %array1007, i32 4
store double 2.500000e-01, double* %elem1012
%elem1013 = getelementptr double, double* %array1007, i32 5
store double 2.500000e-01, double* %elem1013
%elem1014 = getelementptr double, double* %array1007, i32 6
store double 2.500000e-01, double* %elem1014
%elem1015 = getelementptr double, double* %array1007, i32 7
store double 2.500000e-01, double* %elem1015
%elem1016 = getelementptr double, double* %array1007, i32 8
store double 2.500000e-01, double* %elem1016
%elem1017 = getelementptr double, double* %array1007, i32 9
store double 2.500000e-01, double* %elem1017
%elem1018 = getelementptr double, double* %array1007, i32 10
store double 2.500000e-01, double* %elem1018
%elem1019 = getelementptr double, double* %array1007, i32 11
store double 2.500000e-01, double* %elem1019
%elem1020 = getelementptr double, double* %array1007, i32 12
store double 2.500000e-01, double* %elem1020
%elem1021 = getelementptr double, double* %array1007, i32 13
store double 2.500000e-01, double* %elem1021
%elem1022 = getelementptr double, double* %array1007, i32 14
store double 2.500000e-01, double* %elem1022
%elem1023 = getelementptr double, double* %array1007, i32 15
store double 2.500000e-01, double* %elem1023
%elem1024 = getelementptr double, double* %array1007, i32 16
store double 2.500000e-01, double* %elem1024
%elem1025 = getelementptr double, double* %array1007, i32 17
1650  store double 2.500000e-01, double* %elem1025
1651  %elem1026 = getelementptr double, double* %array1007, i32
1652  store double 2.500000e-01, double* %elem1026
1653  %elem1027 = getelementptr double, double* %array1007, i32
1654  store double 2.500000e-01, double* %elem1027
1655  %elem1028 = getelementptr double, double* %array1007, i32
1656  store double 1.250000e+00, double* %elem1028
1657  %actual_list1029 = getelementptr inbounds %list_struct_f,
1658  %list_struct_f* %array_struct1005, i32 0, i32 1
1659  store double* %array1007, double** %actual_list1029
1660  %r4 = alloca %list_struct_f*
1661  store %list_struct_f* %array1007, %list_struct_f** %r4
1662  %theme1030 = load %chordlist_struct*, %chordlist_struct** %theme
1663  %length1031 = getelementptr inbounds %chordlist_struct,
1664  %chordlist_struct* %theme1030, i32 0, i32 0
1665  %size = load i32, i32* %length1031
1666  %r11032 = load %list_struct_f*, %list_struct_f** %r1
1667  %cur_list_ptr = getelementptr inbounds %list_struct_f,
1668  %list_struct_f* %r11032, i32 0, i32 1
1669  %cur_list = load double*, double** %cur_list_ptr
1670  %minor11033 = load %list_struct*, %list_struct** %minor1
1671  %cur_list_ptr1034 = getelementptr inbounds %list_struct,
1672  %list_struct* %minor11033, i32 0, i32 1
1673  %cur_list1035 = load i32*, i32** %cur_list_ptr1034
1674  %length1036 = getelementptr inbounds %list_struct,
1675  %list_struct* %minor11033, i32 0, i32 0
1676  %size1037 = load i32, i32* %length1036
1677  %mallocsize = mul i32 %size, ptrtoint (i32* getelementptr
1678  (i32, i32* null, i32 1) to i32)
1679  %malloccall1038 = tail call i8* @malloc(i32 %mallocsize)
1680  %return_arr = bitcast i8* %malloccall1038 to i32*
1681  %mallocsize1039 = mul i32 %size, ptrtoint (i1**
1682  getelementptr (i1*, i1** null, i32 1) to i32)
1683  %malloccall1040 = tail call i8* @malloc(i32
1684  %mallocsize1039)
1685  %return_arr1041 = bitcast i8* %malloccall1040 to i32**
1686  %mallocsize1042 = mul i32 %size, ptrtoint (i1**
1687  %mallocsize1039)
getelementptr (i1*, i1** null, i32 1) to i32
1679   %malloccall1043 = tail call i8* @malloc(i32
%mallocsize1042)
1680   %norm_arr = bitcast i8* %malloccall1043 to i32***
1681   %length1044 = getelementptr inbounds %chordlist_struct,
%chordlist_struct* %theme1030, i32 0, i32 0
1682   %size1045 = load i32, i32* %length1044
1683   %cur_list_ptr1046 = getelementptr inbounds
%chordlist_struct, %chordlist_struct* %theme1030, i32 0, i32
1
1684   %cur_list1047 = load %chord_struct*, %chord_struct**
%cur_list_ptr1046
1685   %cur_index_ptr = alloca i32
1686   store i32 0, i32* %cur_index_ptr
1687   br label %while
1688  
1689 while: ; preds =
1690   %while_body, %entry
1691   %cur_index2 = load i32, i32* %cur_index_ptr
1692   %pred = icmp ne i32 %size1045, %cur_index2
1693   br i1 %pred, label %while_body, label %merge
1694 while body: ; preds =
1695   %while, %entry
1696   %cur_indexplz = load i32, i32* %cur_index_ptr
1697   %cur_val = getelementptr inbounds %chord_struct,
%chord_struct* %cur_list1047, i32 %cur_indexplz
1698   %val_idx = load %chord_struct, %chord_struct* %cur_val
1699   %cur_val1048 = getelementptr inbounds i32**, i32***
1700   %norm_arr, i32 %cur_indexplz
1701   %stuff = extractvalue %chord_struct %val_idx, 1
1702   %len = getelementptr inbounds i32, i32* %return_arr, i32
%cur_indexplz
1703   %oldlen = extractvalue %chord_struct %val_idx, 0
1704   %len1049 = getelementptr inbounds i32*, i32**
%return_arr1041, i32 %cur_indexplz
1705   %mallocsize1050 = mul i32 %oldlen, ptrtoint (i32*
getelementptr (i32, i32* null, i32 1) to i32)
1706   %malloccall1051 = tail call i8* @malloc(i32
%mallocsize1050)
1707   %clear_cl_list_elem = bitcast i8* %malloccall1051 to i32*
1708   store i32* %clear_cl_list_elem, i32** %len1049
store i32 %oldlen, i32* %len
store i32** %stuff, i32*** %cur_val
%cur_index = load i32, i32* %cur_index_ptr
%new_idx = add i32 %cur_index, 1
store i32 %new_idx, i32* %cur_index_ptr
br label %while

merge: ; preds = %while
%synth-buffer = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 1000))
%synth = call i32 @synth(i32*** %norm_arr, i32 %size, i32* %return_arr, i32 50, i32* %cur_list, i32** %return_arr1, i32 1, i8* %synth-buffer)
%rone = alloca i8*
store i8* %synth-buffer, i8** %rone
%counter1053 = load %chordlist_struct*, %chordlist_struct**
%length1054 = getelementptr inbounds %chordlist_struct, %chordlist_struct* %counter1053, i32 0, i32 0
%size1055 = load i32, i32* %length1054
%r1056 = load %list_struct_f*, %list_struct_f** %r3
%cur_list_ptr1057 = getelementptr inbounds %list_struct_f, %list_struct_f* %r1056, i32 0, i32 1
%cur_list1058 = load double*, double** %cur_list_ptr1057
%minor1059 = load %list_struct*, %list_struct** %minor1
%cur_list_ptr1060 = getelementptr inbounds %list_struct, %list_struct* %minor1059, i32 0, i32 1
%cur_list1061 = load i32*, i32** %cur_list_ptr1060
%length1062 = getelementptr inbounds %list_struct, %list_struct* %minor1059, i32 0, i32 0
%size1063 = load i32, i32* %length1062
%mallocsize1064 = mul i32 %size1065, ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32)
%malloccall1065 = tail call i8* @malloc(i32 %mallocsize1064)
%return_arr1066 = bitcast i8* %malloccall1065 to i32*
%mallocsize1067 = mul i32 %size1065, ptrtoint (i1* getelementptr (i1*, i1** null, i32 1) to i32)
%malloccall1068 = tail call i8* @malloc(i32 %mallocsize1067)
%return_arr1069 = bitcast i8* %malloccall1068 to i32**
%mallocsize1070 = mul i32 %size1055, ptrtoint (i1**
getelemeptr (i1*, i1** null, i32 1) to i32)
%malloccall1071 = tail call i8* @malloc(i32
%mallocsize1070)
%norm_arr1072 = bitcast i8* %malloccall1071 to i32***
%length1073 = getelemeptr inbounds %chordlist_struct,
%chordlist_struct* %counter1053, i32 0, i32 0
%size1074 = load i32, i32* %length1073
%cur_list_ptr1075 = getelemeptr inbounds
%chordlist_struct, %chordlist_struct* %counter1053, i32 0, i32 1
%cur_list1076 = load %chord_struct*, %chord_struct**
%cur_list_ptr1075
%cur_index_ptr1077 = alloca i32
store i32 0, i32* %cur_index_ptr1077
br label %while1078

while1078:
; preds =
%while_body1079, %merge
%cur_index21093 = load i32, i32* %cur_index_ptr1077
%pred1094 = icmp ne i32 %size1074, %cur_index21093
br i1 %pred1094, label %while_body1079, label %merge1095

while_body1079:
; preds =
%while1078
%cur_index1080 = load i32, i32* %cur_index_ptr1077
%cur_val1081 = getelemeptr inbounds %chord_struct,
%chord_struct* %cur_list1076, i32 %cur_index1080
%val_idx1082 = load %chord_struct, %chord_struct*
%cur_val1081
%cur_val1083 = getelemeptr inbounds i32**, i32**
%norm_arr1072, i32 %cur_index1080
%stuff1084 = extractvalue %chord_struct %val_idx1082, 1
%len1085 = getelemeptr inbounds i32, i32*
%return_arr1066, i32 %cur_index1080
%oldlen1086 = extractvalue %chord_struct %val_idx1082, 0
%len1087 = getelemeptr inbounds i32*, i32**
%return_arr1069, i32 %cur_index1080
%malloccsize1088 = mul i32 %oldlen1086, ptrtoint (i32*
getelemeptr (i32, i32* null, i32 1) to i32)
%malloccall1089 = tail call i8* @malloc(i32
%mallocsize1088)
1763   %clear_cl_list_elem1090 = bitcast i8* %malloccall1089 to i32*
1764   store i32* %clear_cl_list_elem1090, i32** %len1087
1765   store i32 %oldlen1086, i32* %len1085
1766   store i32** %stuff1084, i32*** %cur_val1083
1767   %cur_index1091 = load i32, i32* %cur_index_ptr1077
1768   %new_idx1092 = add i32 %cur_index1091, 1
1769   store i32 %new_idx1092, i32* %cur_index_ptr1077
1770   br label %while1078
1771
merge1095:                ; preds = %while1078
1772   %synth-buffer1097 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 1000))
1774   %synth1098 = call i32 @ synth(i32*** %norm_arr1072, i32 %size1055, i32* %return_arr1066, i32 57, i32* %cur_list1061, i32 %size1063, double* %cur_list1058, i32** %return_arr1069, i32 1, i8* %synth-buffer1097)
1775   %rtwo = alloca i8*
1776   store i8* %synth-buffer1097, i8** %rtwo
1777   %theme1099 = load %chordlist_struct*, %chordlist_struct** %theme
1778   %length1100 = getelementptr inbounds %chordlist_struct, %chordlist_struct* %theme1099, i32 0, i32 0
1779   %size1101 = load i32, i32* %length1100
1780   %r21102 = load %list_struct_f*, %list_struct_f** %r2
1781   %cur_list_ptr1103 = getelementptr inbounds %list_struct_f, %list_struct_f* %r21102, i32 0, i32 1
1782   %cur_list1104 = load double*, double** %cur_list_ptr1103
1783   %minor1105 = load %list_struct*, %list_struct** %minor1
1784   %cur_list_ptr1106 = getelementptr inbounds %list_struct, %list_struct* %minor1105, i32 0, i32 1
1785   %cur_list1107 = load i32*, i32** %cur_list_ptr1106
1786   %length1108 = getelementptr inbounds %list_struct, %list_struct* %minor1105, i32 0, i32 0
1787   %size1109 = load i32, i32* %length1108
1788   %mallocsize1110 = mul i32 %size1101, ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32)
1789   %malloccall11111 = tail call i8* @malloc(i32 %mallocsize1110)
%return_arr1112 = bitcast i8* %malloccall1111 to i32*
%mallocsize1113 = mul i32 %size1101, ptrtoint (i1**
getelementptr (i1 *, i1 ** null, i32 1) to i32)
%malloccall1114 = tail call i8* @malloc(i32
%mallocsize1113)
%return_arr1115 = bitcast i8* %malloccall1114 to i32**
%mallocsize1116 = mul i32 %size1101, ptrtoint (i1**
getelementptr (i1 *, i1 ** null, i32 1) to i32)
%malloccall1117 = tail call i8* @malloc(i32
%mallocsize1116)
%norm_arr1118 = bitcast i8* %malloccall1117 to i32***
%length1119 = getelementptr inbounds %chordlist_struct,
%chordlist_struct* %theme1099, i32 0, i32 0
%size1120 = load i32, i32* %length1119
%cur_list_ptr1121 = getelementptr inbounds %chordlist_struct, %chordlist_struct* %theme1099, i32 0, i32 1
%cur_list1122 = load %chord_struct*, %chord_struct**
%cur_list_ptr1121
%cur_index_ptr1123 = alloca i32
store i32 0, i32* %cur_index_ptr1123
br label %while1124
while1124:
  ; preds = %while_body1125, %merge1095
%cur_index21139 = load i32, i32* %cur_index_ptr1123
%pred1140 = icmp ne i32 %size1120, %cur_index21139
br i1 %pred1140, label %while_body1125, label %merge1141
while_body1125:
  ; preds = %while1124
%cur_index21139 = load i32, i32* %cur_index_ptr1123
%cur_val1127 = getelementptr inbounds %chord_struct,
%chord_struct* %cur_list1122, i32 %cur_index21139
%val_idx1128 = load %chord_struct, %chord_struct*
%cur_val1127
%cur_val1129 = getelementptr inbounds i32**, i32***
%norm_arr1118, i32 %cur_index21139
%stuff1130 = extractvalue %chord_struct %val_idx1128, 1
%len1131 = getelementptr inbounds i32, i32*
%return_arr1112, i32 %cur_index21139
%oldlen1132 = extractvalue %chord_struct %val_idx1128, 0
%len1133 = getelementptr inbounds i32*, i32**
%return_arr1115, i32 %cur_indexplz1126
%mallocsize1134 = mul i32 %oldlen1132, ptrtoint (i32* getelementptr (i32, i32* null, i32 1) to i32)
%malloca111135 = tail call i8* @malloc(i32 %mallocsize1134)
%clear_cl_list_elem1136 = bitcast i8* %malloca111135 to i32*
store i32* %clear_cl_list_elem1136, i32** %len1133
store i32 %oldlen1132, i32* %len1131
store i32** %stuff1130, i32*** %cur_val1129
%cur_index1137 = load i32, i32* %cur_index_ptr1123
%new_idx1138 = add i32 %cur_index1137, 1
store i32 %new_idx1138, i32* %cur_index_ptr1123
br label %while1124

merge1141: ; preds = %while1124
%synth-buffer1143 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 1000))
%synth1144 = call i32 @synth(i32*** %norm_arr1118, i32 %size1101, i32* %return_arr1112, i32 33, i32* %cur_list1107, i32 %size1109, double* %cur_list1104, i32** %return_arr1115, i32 2, i8* %synth-buffer1143)
%lone = alloca i8*
store i8* %synth-buffer1143, i8** %lone
%new_string = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 1000))
%first = getelementptr inbounds i8, i8* %new_string, i32 0
store i8 0, i8* %first
%rone1146 = load i8*, i8** %rone
%put_first = call i8* @strcat(i8* %new_string, i8* %rone1146)
%lone1147 = load i8*, i8** %lone
%put_first1148 = call i8* @strcat(i8* %new_string, i8* %lone1147)
%song = alloca i8*
store i8* %new_string, i8** %song
%new_string1150 = tail call i8* @malloc(i32 mul (i32 ptrtoint (i8* getelementptr (i8, i8* null, i32 1) to i32), i32 1000))
%first1151 = getelementptr inbounds i8, i8* %new_string1150, i32 0
store i8 0, i8* %first1151
%song1152 = load i8*, i8** %song
%put_first1153 = call i8* @strcat(i8* %new_string1150, i8* %song1152)
%rtwo1154 = load i8*, i8** %rtwo
%put_first1155 = call i8* @strcat(i8* %new_string1150, i8* %rtwo1154)
%song1 = alloca i8*
store i8* %new_string1150, i8** %song1
%song11156 = load i8*, i8** %song1
%make_midi = call i32 @make_midi(i8* %song11156, i8*
  getelementptr inbounds ([10 x i8], [10 x i8]* @0, i32 0, i32 0))
ret i32 0
}

declare i32 @synth(i32***, i32, i32*, i32, i32*, i32, double*, i32**, i32, i8*)

declare i8* @strcat(i8*, i8*)

declare i32 @make_midi(i8*, i8*)

declare i32 @malloc(i32)
y = 6;
def Fun x = x;
z = Fun(y);
Printint(z);
@fmt = private unnamed_addr constant [4 x i8] c"%d\0A\00"
@str = private unnamed_addr constant [4 x i8] c"%s\0A\00"
@flt = private unnamed_addr constant [4 x i8] c"%f\0A\00"
@str.1 = private unnamed_addr constant [3 x i8] c"%c\00"
@fmt.2 = private unnamed_addr constant [4 x i8] c"%d \00"
@fmt.3 = private unnamed_addr constant [4 x i8] c"%f \00"

declare i32 @printf(i8*, ...)
define i32 @main() {
  entry:
  %y = alloca i32
  store i32 6, i32* %y
  %y1 = load i32, i32* %y
  %Fun = call i32 @Fun(i32 %y1)
  %z = alloca i32
  store i32 %Fun, i32* %z
  %z2 = load i32, i32* %z
  %printf = call i32 (i8*, ...) @printf(i8* getelementptr
  inbounds ([4 x i8], [4 x i8]* @fmt, i32 0, i32 0), i32 %z2)
  ret i32 0
}
declare i32 @synth(i32***, i32, i32*, i32, i32*, i32, double*,
i32**, i32, i8*)
declare i8* @strcat(i8*, i8*)
declare i32 @make_midi(i8*, i8*)
define i32 @Fun(i32 %x) {
  entry:
  %x1 = alloca i32
  store i32 %x, i32* %x1
  %x2 = load i32, i32* %x1
  ret i32 %x2
}
6.2 Test Suite
fail-arith-1.mike
3 a=1+.1;
fail-arith2.mike
6 1.0+1.0;
fail-array2.mike
11 arr=[ "c" 2 3 4 ];
fail-array.mike
15 arr=[1 1. 3 ];
fail-assign.mike
18 a=c+1;
fail-if1.mike
23 if true then 5;
fail-if.mike
28 if 42 then Printint(5) else Printint(3);
fail-illegal-id.mike
31 b=2;
fail-index.mike
36 a=[1 2 3 ];
            b=a.[ 3. ];
fail-pitch-empty.mike
40 p:[3 4 -4 ];
/* pitch literals cannot be negative */

test-1synth1.mike

test-array.mike

test-assign-array.mike
83
84  test-assign.mike
85
86  a = 5;
87  Printint (a);
88
89  test-assign-print.mike
90
91
92
93  a = 5; Printint(a);
94
95  test-block-return.mike
96
97  /* testing whether {4;5;10;} returns "10" as its value */
98  Printint({4;5;10;});
99
100  test-channel-synth.mike
101
102  p1=p:[ 1 2 3 4 5];
103  model1= [ 1 2 3 4 5];
104  rlist1=r:[1 1 1 1 1];
105  start_note1=50;
106  channel1=0;
107
108
109
110  test-flat-v.mike
111
112  p:[ v4 v4 6b|4#];
113
114  test-floatlist.mike
115
116  [ 1. 1. 1. 1. ];
117
118  test-float.mike
119
120  Printfloat(1.2345);
121
122  test-func-decl-call.mike
123
124
def Name = {Printint(10); Printstr("hello"); a = 5 + 10; a;};
Printint(Name());

def Name hello = {Printint(10); Printstr("hello"); a = 5 + 10; a;};

test-if.mike
if true then Printint(5) else Printint(10);
if false then Printint(5) else Printint(10);
Printint(15);

test-index2.mike
a=[2 3 4 ];
c=a.[0+1];
Printint(c);

test-index-print.mike
a = [1 2 3]; Printint(a.[2]);

test-int-add.mike
Printint(1 + 10);

test-int.mike
Printint(5);

test-map-1.mike
l = [1 2 3 4 ];
def A a = a + 2;
Map( A l);
test-pitch-empty.mike
p: [ ^3# ^5# ^8# ];
test-printadd.mike
Printint(5 + 5);
test-printFadd.mike
Printfloat(5. + 5.);
test-print-index2.mike
a = [ 2 3 4 5 ];
c = a. [0+1];
Printint(c);
test-printint.mike
Printint(5);
test-print-list.mike
l = [ 1 2 3 4 5 ];
Printlist( l);
test-rdot.mike
r: [ wo 1. 4.5 8o ];
test-rhythm-empty.mike
r: [ w 1. 3. h q ];
test-rhythm-list-dec.mike
a = r: [ q w ];
208  Printrlist(a);
209
210  test-scope.mike
211
212  {
213      a=50;
214      Printint(a);
215  };
216  Printint(a);
217
218  test-string.mike
219
220  Printstr("helloworld");
221
222  test-subset.mike
223
224  a=[ 2 3 4 ];
225  c=a.[2];
226  Printint(c);
227
228  test-type.mike
229
230  /* assume print is not different words */
231  a="hello";
232  Print( a);
233  b=8;
234  Print(b);
235  c='c';
236  Print(c);
237  d=[1 2 3];
238  Print(d);
239
240  test-unary.mike
241
242  a= -1;
243  PrintInt a;
244
245  fail-pitch-empty.out
246
247
248  test-1synth1.out
V1 [61]/1.00 [63]/1.00 [65]/1.00 [66]/1.00 [65]/1.00 [63]/1.00 [61]/1.00 V2 [64]/1.00 [66]/1.00 [68]/1.00 [69]/1.00 [68]/1.00 [66]/1.00 [64]/1.00

test-arith1.out

4
8
16
5
test-arith2.out

2.220000
5.500000
8.000000
32.000000
3.500000
test-array.out

2.220000
5.500000
8.000000
32.000000
3.500000
test-assign-array.out

5
test-assign.out

5
test-assign-print.out

5
test-block-return.out

10
test-flat-v.out

test-flatlist.out
290
test-float.out
292
293 1.234500
294
test-func-decl-call.out
296
297 10
298 hello
299 15
300
test-func-decl.out
302
303
test-if.out
305
306 5
307 10
308 15
309
test-index2.out
311
312 3
313
test-index-print.out
315
316 3
317 test-int-add.out
318
319 11
320
test-int.out
322
323 5
324
test-map-1.out
326
327 4 4 4 4
328
test-pitch-empty.out
330
331
332  test-printadd.out
333
334  10
335
336  test-printFadd.out
337
338  10.000000
339
340  test-print-index2.out
341
342  3
343
344  test-printint.out
345
346  5
347
348  test-print-list.out
349
350  1 2 3 4 5
351
352  test-rdot.out
353
354
355  test-rhythm-empty.out
356
357
358  test-rhythm-list-dec.out
359
360  1.000000 4.000000
361
362  test-scope.out
363
364  50
365
366  test-string.out
367
368  helloworld
369
370  test-subset.out
371
372  4
373
test-type.out
hello
8
fail-arith-1.err
Fatal error: exception Failure("Mismatched types")
fail-arith2.err
Fatal error: exception Failure("Mismatched types")
fail-array2.err
Fatal error: exception Failure("Mismatched types")
fail-array.err
Fatal error: exception Failure("Mismatched types")
fail-assign.err
Fatal error: exception Failure("c Not Found")
fail-assign-print.err
Fatal error: exception Failure("a Not Found")
fail-if1.err
Fatal error: exception Parsing.Parse_error
fail-if.err
Fatal error: exception Failure("Mismatched types")
fail-illegal-id.err
Fatal error: exception Parsing.Parse_error
fail-index.err
Fatal error: exception Parsing.Parse_error
6.3 Test Suites

We wrote tests for every time we implemented a new feature. We also wrote integration tests to make sure that our modules worked in conjunction with each other. For each test, we also wrote an expected output file.

6.4 Test Automation

Our test automation script, testall.sh, runs every test and compares it to the expected output for that test. The testing script is included in the Appendix.
7. Lessons Learned

7.1 Team Reflections

7.1.1 Harvey Wu

This was, by far, the largest project of my programming career, and the first one that involved a lot of teamwork. I learned to communicate often with my team, even if not much progress was being made, if just to touch base and keep each other interested in the project. Safe to say, I also got to know the ins-and-outs of Git pretty well.

I had never touched functional programming before this class and I loved the ”if it compiles it works” aspect of OCaml. By implementing a type inference algorithm I also learned a lot about what makes OCaml tick.

Our group spent too much time early on arguing about syntax and design choices that seemed relatively trivial down the road. For future groups: decide on something, and stick to it. Unless it breaks :)
7.1.2 Kaitlin Pet

For the last three years, I have had two reach classes at Columbia- ones whose content I found fascinating but seemed extremely challenging. PLT was one of those classes, am Im glad I finally got to take it as a second semester senior. Because I am concentrating in computer science, this was my first upper-level project-based course and the second major project Ive done at Columbia (the first being a genetic engineering team-based composition that had little to do with computer science). Learning OCaml was an overall fun experience, as someone who hasnt taken Fundamentals the LLVM aspect was a lot more complicated. If music grad school doesn’t work out Id definitely like to do more with functional programming. I also liked that you need very little prior information coming in; starting the class I felt on the same page as everyone else. Lastly Id like to thank my teammates, Ive head a lot of horror stories and had a really great experience code grinding with these guys. To future teams, I would suggest 1) bonding with your team early on to feel a sense of connection/obligation to each other, and 2) considering including concentrators because we have hella more time on our hands.

7.1.3 Lakshmi Bodapati

I learned a lot about Git and pull requests and how easily git branches can become messy, unorganized and complicated tree of merge conflicts. I also learned how to solve those problems, maybe not in the Git recommended
way, but some way of getting a working master branch after merging in 4+ branches of conflicting code. I also learned a lot about dividing work and maximizing efficiency when all of us were working together and trying to knock out chunks of the project during our longer weekly meetings. The trickiest part was dividing up work such that everyone had something to do that wasn’t blocked by something someone else had to do and every task was being worked on by someone who had some idea of what was necessary for successful completion. The coolest thing I learned was how to write a context-sensitive scanner. I learned the most OCaml while writing the polymorphic function definitions and call matching in the semant module. I also learned that OCaml is quite frosty and nice. For future teams, start early, own features not modules and don’t be afraid to OCaml it up!

7.1.4 Husam Abdul-Kafi

I learned a lot about how coding a big group project with a lot of people with varying levels of time and preparedness. It was interesting to work around everyone’s schedule and maximize the usage of our time. I love OCaml, and I have no problems with using it for this project. I did a past project where we compiled to a version of Java IR (I think it was called Jade?). That project didn’t use anything like the OCAML LLVM Model in that we had to emit every line of code in the IR representation by hand. It was really interesting to compare the two types of IR representation (Java JVM is stack based) and
learn the differences.

Obviously, I’d recommend any future project to have a working version at least a week before the final demo so you can spend time optimizing your code and getting edge cases. Don’t forget edge cases!!
8. Appendix
(* Abstract Syntax Tree and functions for printing it *)

type op = Add | FAdd | Sub | FSub | Mul | FMul | Div | FDiv
          | Equal | Neq | Less | Leq | Greater | Geq | And | Or

type preop = Neg | Not | FNeg

type postop = Rhythmdot

type typ =
          TUnit
          | TInt
          | TBool
          | TFloat
          | TStr
          | TPitch
          | TChord
          | TType of string
          | TList of typ
          | TFun of typ list * typ

type expr =
          Literal of int
          | FloatLit of float
          | BoolLit of bool
          | ID of string
          | String of string
          | Binop of expr * op * expr
          | Preop of preop * expr
          | Postop of expr * postop
          | Assign of string * expr
          | Call of expr * expr list
          | If of expr * expr * expr
          | Pitch of int * expr * int
          | Chord of expr list
          | Subset of expr * expr
          | List of expr list
          | PList of expr list
          | ChordList of expr list (*PList --> "list of chords")
          | RList of expr list
type aexpr =
  ALiteral of int * typ
| AFloatLit of float * typ
| ABoolLit of bool * typ
| AID of string * typ
| AString of string * typ
| ABinop of aexpr * op * aexpr * typ
| APreop of preop * aexpr * typ
| APPostop of aexpr * postop * typ
| AAssign of string * aexpr * typ
| ACall of aexpr * aexpr list * typ
| AIf of aexpr * aexpr * aexpr * typ
| ASubset of aexpr * aexpr * typ
| AList of aexpr list * typ
| APList of aexpr list * typ
| APitch of int * aexpr * int * typ
| AChord of aexpr list * typ
| AChordList of aexpr list * typ (*PList \rightarrow "list of chords"*)
| ARList of aexpr list * typ
| ABlock of aexpr list * typ
| AConcat of aexpr * aexpr * typ
| ANoexpr
| AFun of string * string list * aexpr * typ (* might be some issue with formals as string list *)
| AUnit of typ

type program = expr list

let string_of_op = function
  Add -> "+"
| FAdd -> "+."

(* Pretty-printing functions *)
let string_of_preop = function
  Neg -> "-"
  Not -> "!
  FNeg -> "-"
(* OctaveUp -> "^" *)
OctaveDown -> "v" *)
let string_of_postop = function
  Sharp -> "#"
  Flat -> "b" *)
  Rhythmdot -> "o"
(* Postop(e, o) -> string_of_expr e ^ string_of_postop o *)
let rec string_of_expr = function
  Literal(l) -> string_of_int l
  FloatLit(f) -> string_of_float f
  Boollit(true) -> "true"
  Boollit(false) -> "false"
  ChordList(_ _) -> "Chordlist"
  ID(s) -> s
  String(s) -> s
  Binop(e1, o, e2) ->
    string_of_expr e1 ^ " " ^ string_of_op o ^ " " ^
    string_of_expr e2
  Preop(o, e) -> string_of_preop o ^ string_of_expr e
  (* Postop(e, o) -> string_of_expr e ^ string_of_postop o *)
  Assign(v, e) -> "Assign( " ^ v ^ " = " ^ (string_of_expr e)
^ "})" 
121 | Call(f, el) -> 
122   string_of_expr f "(" String.concat "", " (List.map string_of_expr el) ")" 
123 | If(e1, e2, e3) -> "if " String.concat " then " String.concat " else " String.concat ".
124 | Subset(s, i) -> string_of_expr s "].[" String.concat " "] ^ "]
125 | List(es) -> "[ " String.concat " (List.map string_of_expr es) "]"
126
127 (* | ChordList(cs) -> 
128   let string_of_chord ps = 
129     let string_of_pitch (i1, e, i2) = 
130       if i1 < 0 && i2 < 0 then 
131       (String.make (abs i1) 'v') String.concat (string_of_expr e) ^ 
132      (String.make (abs i2) 'b') 
133     else 
134       if i1 < 0 && i2 >= 0 then 
135       (String.make (abs i1) 'v') String.concat (string_of_expr e) ^ 
136      (String.make (abs i2) '#') 
137     else 
138       if i1 >= 0 && i2 < 0 then 
139       (String.make (abs i1) '^') String.concat (string_of_expr e) ^ 
140      (String.make (abs i2) 'b') 
141     else 
142       (String.make (abs i1) '^') String.concat (string_of_expr e) ^ 
143      (String.make (abs i2) '#') 
144 in 
145     String.concat "|" (List.map string_of_pitch ps) 
146 in 
147   "p:[" String.concat " (List.map string_of_chord cs) 
148   "]" *)
149
150 | RList(es) -> "r:[ " String.concat " (List.map string_of_expr es) "]"
151 | Block(es) -> "{ " String.concat " (List.map string_of_expr es) "}"
152 | Concat(e1, e2) -> string_of_expr e1 "@" string_of_expr e2 
153 | Noexpr -> ""

147
150  Unit -> "()"
151  Fun(n, args, e) -> "Fun " ^ n ^ String.concat " " args ^ " = " ^ string_of_expr e
152  _ -> "string_of_expr not implemented for your expression yet."
153
154  let rec string_of_typ = function
155       TInt -> "[int]"
156       TBool -> "[bool]"
157       TFloat -> "[float]"
158       TString -> "[string]"
159       TPitch -> "[pitch]"
160       TUnit -> "[unit]"
161       TType(s) -> "[" ^ s ^ "]"
162       TFun(t1, t2) -> String.concat " " (List.map string_of_typ t1) ^ " " ^ string_of_typ t2
163       TList(s) -> string_of_typ s ^ "list"
164
165  let rec string_of_aexpr = function
166       ALiteral(l,t) -> string_of_int l ^ string_of_typ t
167       AFloatLit(f,t) -> string_of_float f ^ string_of_typ t
168       ABoolLit(true, t) -> "true" ^ string_of_typ t
169       ABoolLit(false, t) -> "false" ^ string_of_typ t
170       AID(s, t) -> s ^ string_of_typ t
171       AString(s, t) -> s ^ string_of_typ t
172       APitch(i1, ae, i2, t) ->
173       let signs = (i1 >= 0, i2 >= 0) in
174       match signs with
175       | false, false -> (String.make (abs i1) '^') ^ (string_of_aexpr ae) ^ (String.make (abs i2) '#')
176       | false, true -> (String.make (abs i1) '^') ^ (string_of_aexpr ae) ^ (String.make (abs i2) 'v')
177       | true, false -> (String.make (abs i1) 'v') ^ (string_of_aexpr ae) ^ (String.make (abs i2) '^')
178       | _ -> (String.make (abs i1) '^') ^ (string_of_aexpr ae) ^ (String.make (abs i2) '#')
let string_of_program (exprs) =
"EXPRESSIONS: " ^ String.concat "\n" (List.map string_of_expr exprs) ^ "\n"

let string_of_inferred (aexprs) =
"INFERRED EXPRS: " ^ String.concat "\n" (List.map string_of_aexpr aexprs) ^
"\n"
(*
Code generation: translate takes a semantically checked AST and produces LLVM IR
Detailed documentation on the OCaml LLVM library:
http://llvm.moe/
http://llvm.moe/ocaml/
*)

module L = LLVM
module A = Ast
module I = Infer

module StringMap = Map.Make(String)

let first (a,_,_) = a;
let second (_,a,_) = a;
let third (_,_,a) = a;

let main_vars:(string, L.llvalue) Hashtbl.t = Hashtbl.create 100
let function_defs:(string, (L.llvalue * A.aexpr)) Hashtbl.t = Hashtbl.create 100

(*, functions, structs *)
let translate (exprs) =
  let context = L.global_context () in
  let names:(string, L.llvalue) Hashtbl.t = Hashtbl.create 10 in
  let the_module = L.create_module context "MusicMike"
  and i32_t = L.i32_type context (* integer *)
  and i8_t = L.i8_type context (* char? *)
  and il_t = L.il_type context (* boole *)
  and float_t = L.double_type context (* float *)
  and void_t = L.void_type context in (* void *)
  let i8p_t = L.pointer_type i8_t in (* char pointer-string*)
let i32p_t = L.pointer_type i32_t in (* int* *)
let i32pp_t = L.pointer_type i32p_t in (* int** *)
let i32ppp_t = L.pointer_type i32pp_t in (* int*** *)
let floatp_t = L.pointer_type float_t in (* float* *)

(* int list struct *)
let list_t = L.named_struct_type context "list_struct" in
L.struct_set_body list_t [| i32_t ; i32p_t |] true;
let listp_t = L.pointer_type list_t in

(* int * list struct *)
let chord_struct = L.named_struct_type context "chord_struct" in
L.struct_set_body chord_struct [| i32_t ; i32pp_t |] true;
let chord_structp = L.pointer_type chord_struct in

(* int ** list struct *)
let chordlist_struct = L.named_struct_type context "chordlist_struct" in
L.struct_set_body chordlist_struct [| i32_t ; chord_structp |] true;
let chordlist_structp = L.pointer_type chordlist_struct in

(* float list struct *)
let list_t_f = L.named_struct_type context "list_struct_f" in
L.struct_set_body list_t_f [| i32_t ; floatp_t |] true;
let listp_t_f = L.pointer_type list_t_f in

let ltype_of_typ = function
  | A.TInt    -> i32_t
  | A.TBool   -> i1_t
  | A.TList(A.TInt)   -> listp_t
  | A.TList(A.TFloat) -> listp_t_f
  | A.TList(A.TList(A.TPitch)) -> chordlist_structp
  | A.TVoid    -> void_t *
  | A.TFloat   -> float_t
  | A.TString  -> i8p_t
  | A.TUnit    -> void_t
  | t          -> raise (Failure (A.string_of_typ(t) ^ "Shouldn't be here")) in

let stype_of_typ = function
  A.TList(A.TInt) -> (i32_t, list_t)
A.TList(A.TFloat) -> (float_t, list_t_f)
_ -> raise (Failure "No Struct of this type")

(* Declare printf(), which the print built-in function will call *)
let printf_t = L.var_arg_function_type i32_t []
L.pointer_type i8_t [] in
let printf_func = L.declare_function "printf" printf_t
the_module in
let default_fun = L.define_function "main" (L.function_type (ltype_of_typ A.TInt) [[]]) the_module in
let builder = L.builder_at_end context (L.entry_block default_fun) in
let int_format_str = L.build_global_stringptr "%d
" "fmt"
builder in
let str_format = L.build_global_stringptr "%s
" "str"
builder in
let float_format = L.build_global_stringptr "%f
" "flt"
builder in
let char_no_line = L.build_global_stringptr "%c" "str"
builder in
let int_no_line = L.build_global_stringptr "%d " "fmt"
builder in
let float_no_line = L.build_global_stringptr "%f " "fmt"
builder in

(* Declare the built-in synth() function *)
let synth_t = L.function_type i32_t [] [i32ppp_t; i32_t; i32p_t; i32p_t; i32_t; i32p_t; i32pp_t; i32_t; i8p_t []]
in
let synth_func = L.declare_function "synth" synth_t
the_module in

(* Declare the build-in make_midi() function*)
let make_midi_t = L.function_type i32_t [] [i8p_t; i8p_t] in
let make_midi_func = L.declare_function "make_midi"
make_midi_t the_module in
let get_length (struct_obj, sub_builder) = (* get pointer to length in the struct (at position 0,0) *)
  pointer = L.build_in_bounds_gep struct_obj [ L.const_int i32_t 0; L.const_int i32_t 0 ] "length" sub_builder in
  (* load that pointer to the length *)
  L.build_load pointer "size" sub_builder in

let get_list (struct_obj, sub_builder) = (* get pointer to the int* in the struct (at position 0,1) *)
  list_pointer = L.build_in_bounds_gep struct_obj [ L.const_int i32_t 0; L.const_int i32_t 1 ] "cur_list_ptr" sub_builder in
  (* load that pointer - now act_list is the pointer to the head of the list *)
  L.build_load list_pointer "cur_list" sub_builder in

(* s_list is llvalue, application is function taking element of list, index, and builder *)
let map s_list application = (* get pointer to length in the struct (at position 0,0) *)
  pointer = L.build_in_bounds_gep s_list [ L.const_int i32_t 0; L.const_int i32_t 0 ] "length" builder in
  (* load that pointer to the length *)
  let length = L.build_load pointer "size" builder in
  (* get pointer to the int* in the struct (at position 0,1) *)
  list_pointer = L.build_in_bounds_gep s_list [ L.const_int i32_t 0; L.const_int i32_t 1 ] "cur_list_ptr" builder in
  (* load that pointer - now act_list is the pointer to the head of the list *)
  act_list = L.build_load list_pointer "cur_list" builder in
  (* allocate a pointer to an int (on the stack) *)
  cur_index_ptr = L.build_alloca i32_t "cur_index_ptr" builder in
(* store a 0 in that location *)
let cur_index = L.build_store (L.const_int i32_t 0)
cur_index_ptr builder in

(* we are creating blocks, so we need the function
we are currently in *)
let cur_fun = L.block_parent (L.insertion_block builder) in

(* create the block that's supposed to have
"the conditional block" => pred_bb *)
let pred_bb = L.append_block context "while"
cur_fun in
ignore (L.build_br pred_bb builder);

(* create the block of the body - basically
printf act_list[cur_index] *)
let body_bb = L.append_block context "while_body"
cur_fun in

(* body_builder is the builder in the "while
body" *)
let body_builder = L.builder_at_end context
body_bb in

(* DO THE WORK ON THE ACTUAL ELEMENTS OF THE LIST
HERE *)

(* loads the value in cur_index_ptr *)
let cur_idx_in_body = L.build_load cur_index_ptr "cur_indexplz" body_builder in

(* get a pointer into the list at the index
with the value just loaded *)
let ptr_to_idx = L.build_in_bounds_gep act_list [ cur_idx_in_body | ] "cur_val" body_builder in

(* load the value at that pointer (aka value
of act_list[cur_index]) *)
let val_idx = L.build_load ptr_to_idx "val_idx" body_builder in

(* apply function onto element*)
ignore(application val_idx cur_idx_in_body
body_builder);

(* print_endline("line 169");*)
(* END WORK HERE *)

(* loads the value in cur_index_ptr *)
let cur_index_val = L.build_load cur_index_ptr "cur_index" body_builder in
(* add 1 to the value *)
let new_idx = L.build_add cur_index_val (L.const_int i32_t 1) "new_idx" body_builder in
(* store the new value in the pointer *)
ignore(L.build_store new_idx cur_index_ptr body_builder);
ignore(L.build_br pred_bb body_builder);

(* the builder at the "check if cur_index < length" *)
let pred_builder = L.builder_at_end context pred_bb in
let cur_index_val2 = L.build_load cur_index_ptr "cur_index2" pred_builder in
let bool_val = L.build_icmp L.Icmp.Ne length cur_index_val2 "pred" pred_builder in

let merge_bb = L.append_block context "merge" cur_fun in
ignore(L.build_cond_br bool_val body_bb merge_bb pred_builder);
L.position_at_end merge_bb builder;
in

let rec expr builder = function
  A.ALiteral(i, _) -> L.const_int i32_t i
  | A.AFloatLit(f, _) -> L.const_float float_t f
  | A.ABoolLit(b, _) -> L.const_int i1_t (if b then 1 else 0)
  | A.ANoexpr -> L.const_int i32_t 0
  | A.AID(s, _) -> L.build_load (try Hashtbl.find main_vars s with Not_found -> raise(Failure(s ^ " Not Found"))) s builder
  | A.AString(s, _) -> L.build_global_stringptr s "" builder
  | A.ABinop (e1, op, e2, _) ->
    let e1' = expr builder e1
    and e2' = expr builder e2 in
    (match op with
      A.Add -> L.build_add
      | A.Sub -> L.build_sub
      | A.Mult -> L.build_mul
      | A.Div -> L.build_sdiv
| 194  | A.FAdd    -> L.build_fadd                  |
| 195  | A.FSub    -> L.build_fsub                 |
| 196  | A.FMult   -> L.build_fmul                 |
| 197  | A.FDiv    -> L.build_fdiv                 |
| 198  | A.And     -> L.build_and                  |
| 199  | A.Or      -> L.build_or                   |
| 200  | A.Equal   -> L.build_icmp L.Icmp.Eq       |
| 201  | A.Neq     -> L.build_icmp L.Icmp.Ne       |
| 202  | A.Less    -> L.build_icmp L.Icmp.Slt      |
| 203  | A.Greater -> L.build_icmp L.Icmp.Sgt      |
| 204  | A.Geq     -> L.build_icmp L.Icmp.Sge      |
| 205  | A.Pitch   -> ( preop, e, postop, _ ) ->   |
| 206  | (* allocates single pitch *)              |
| 207  | let arr_pitch_malloc = L.build_array_malloc i32_t (L.const_int i32_t 3) "array" builder in |
| 208  | (* prefield *)                            |
| 209  | let pref_field_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 0)] "prefield_elem" builder in |
| 210  | let el' = L.const_int i32_t preop in       |
| 211  | ignore(l.build_store el' pref_field_pointer builder); |
| 212  | (* scale degree *)                        |
| 213  | let sd_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 1)] "scaleddegree_elem" builder in |
| 214  | let el' = expr builder e in               |
| 215  | ignore(l.build_store el' sd_pointer builder); |
| 216  | (* posfield *)                            |
| 217  | let post_field_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 2)] "postfield_elem" builder in |
| 218  | let el' = L.const_int i32_t postop in      |
| 219  | ignore(l.build_store el' post_field_pointer builder); |
| 220  | arr_pitch_malloc                          |
| 221  | A.List (es, t) -> ( match t with          |
| 222  | | TList(TList(TPitch)) ->                   |
| 223  | (* allocate struct to hold chordlist and length, |
| 224  | struct_c { i32_t, struct_b* } *)          |
| 225  | let cl_struct = L.build_malloc chordlist_struct |
| 226  | "cl_struct" builder in                    |
| 227  | A.Pitch (preop, e, postop, _ ) ->         |
| 228  | (* allocates single pitch *)              |
| 229  | let arr_pitch_malloc = L.build_array_malloc i32_t (L.const_int i32_t 3) "array" builder in |
| 230  | (* prefield *)                            |
| 231  | let pref_field_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 0)] "prefield_elem" builder in |
| 232  | let el' = L.const_int i32_t preop in       |
| 233  | ignore(l.build_store el' pref_field_pointer builder); |
| 234  | (* scale degree *)                        |
| 235  | let sd_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 1)] "scaleddegree_elem" builder in |
| 236  | let el' = expr builder e in               |
| 237  | ignore(l.build_store el' sd_pointer builder); |
| 238  | (* posfield*)                             |
| 239  | let post_field_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 2)] "postfield_elem" builder in |
| 240  | let el' = L.const_int i32_t postop in      |
| 241  | ignore(l.build_store el' post_field_pointer builder); |
| 242  | arr_pitch_malloc                          |
| 243  | A.List (es, t) -> ( match t with          |
| 244  | | TList(TList(TPitch)) ->                   |
| 245  | (* allocate struct to hold chordlist and length, |
| 246  | struct_c { i32_t, struct_b* } *)          |
| 247  | let cl_struct = L.build_malloc chordlist_struct |
| 248  | "cl_struct" builder in                    |
| 249  | A.Pitch (preop, e, postop, _ ) ->         |
| 250  | (* allocates single pitch *)              |
| 251  | let arr_pitch_malloc = L.build_array_malloc i32_t (L.const_int i32_t 3) "array" builder in |
| 252  | (* prefield *)                            |
| 253  | let pref_field_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 0)] "prefield_elem" builder in |
| 254  | let el' = L.const_int i32_t preop in       |
| 255  | ignore(l.build_store el' pref_field_pointer builder); |
| 256  | (* scale degree *)                        |
| 257  | let sd_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 1)] "scaleddegree_elem" builder in |
| 258  | let el' = expr builder e in               |
| 259  | ignore(l.build_store el' sd_pointer builder); |
| 260  | (* posfield*)                             |
| 261  | let post_field_pointer = L.build_gep arr_pitch_malloc [| (L.const_int i32_t 2)] "postfield_elem" builder in |
| 262  | let el' = L.const_int i32_t postop in      |
| 263  | ignore(l.build_store el' post_field_pointer builder); |
| 264  | arr_pitch_malloc                          |
| 265  | A.List (es, t) -> ( match t with          |
| 266  | | TList(TList(TPitch)) ->                   |
| 267  | (* allocate struct to hold chordlist and length, |
| 268  | struct_c { i32_t, struct_b* } *)          |
| 269  | let cl_struct = L.build_malloc chordlist_struct |
| 270  | "cl_struct" builder in                    |
(* builds pointer for length field in struct to hold length of chordlist *)

let c_len_pointer = L.build_in_bounds_gep cl_struct [ | L.const_int i32_t 0; L.const_int i32_t 0 | ] "length" builder in

ignore(L.build_store (L.const_int i32_t (List.length es)) c_len_pointer builder);

(* malloc's an array of type struct_b {i32_t, i32_t**} *)

let arr_malloc = L.build_array_malloc chord_struct (L.const_int i32_t (List.length es)) "chord_pointer_array" builder in

(* makes chord struct- length + content *)

(* iterates thru es and builds each chord *)

let iter_thru_chord index chord =

let cl2c_pointer = L.build_gep arr_malloc [| (L.const_int i32_t index) || ] "pointer_chord_elem_list" builder in

let e' = expr builder chord in

let e_val = L.build_load e' "actual_chord_struct" builder in

ignore(L.build_store e_val cl2c_pointer builder);

in

(* iterates through chords with iter_thru_chord *)

List.iteri iter_thru_chord es;

(* make pointer to chord array in s list *)

let cl_pointer_arr = L.build_in_bounds_gep cl_struct [| L.const_int i32_t 0; L.const_int i32_t 1 | ] "struct_cl_pointer" builder in

(* fill arr_malloc into pointer to chord list struct *)

ignore(L.build_store arr_malloc cl_pointer_arr builder); cl_struct

| TList(TPitch) -> (*A List of Pitches (aka things separated by |) *) *

(* allocates the chord list *)

let c_struct = L.build_malloc chord_struct "chord_struct" builder in

let c_len_pointer = L.build_in_bounds_gep c_struct [| L.const_int i32_t 0; L.const_int i32_t 0 | ]
"length" builder in
257  ignore(L.build_store (L.const_int i32_t
(List.length es)) c_len_pointer builder);
258  let arr_chord_malloc = L.build_array_malloc i32p_t
(L.const_int i32_t (List.length es)) "arr_pitch" builder in
259  (* ties c_struct to cl_struct *)
260  let deal_with_pitch index el =
261  (* assigns a pointer to the pitch *)
262  let pitch_pointer = L.build_gep arr_chord_malloc
([], (L.const_int i32_t index)) "pitch_pointer_elem" builder in
263  (* let e_val = L.build_load e'
264  "actual_pitch_struct" builder in *)
265  ignore(L.build_store e' pitch_pointer builder);
266  in
267  (* iterates through pitches with deal_with_pitch *)
268  List.iteri deal_with_pitch es;
269  let c_pointer_arr = L.build_in_bounds_gep c_struct
([], L.const_int i32_t 0; L.const_int i32_t 1 [])
"struct_c_pointer" builder in
270  ignore(L.build_store arr_chord_malloc
c_pointer_arr builder); c_struct
271  [ ] ->
272  let s_list = L.buildalloca (snd (stype_of_typ t))
"array_struct" builder in
273  let pointer = L.build_in_bounds_gep s_list [],
L.const_int i32_t 0; L.const_int i32_t 0 [] "length" builder in
274  ignore(L.build_store (L.const_int i32_t
(List.length es)) pointer builder);
275  let arr_alloc = L.build_arrayalloca (fst
(stype_of_typ t)) (L.const_int i32_t (List.length es)) "array" builder
276  in
277  let deal_with_element index e =
278  let pointer = L.build_gep arr_alloc [],
(L.const_int i32_t index)]) "elem" builder in
279  let e' = expr builder e in
280  ignore(L.build_store e' pointer builder)
281  in List.iteri deal_with_element es;
282  let pointer_arr = L.build_in_bounds_gep s_list
([], L.const_int i32_t 0; L.const_int i32_t 1 []) "actual_list" builder in

159
ignore(L.build_store arr_alloc pointer_arr builder);  s_list

| A.ASubset(e1, e2, _) ->
| let s_list = expr builder e1 in
| let index = expr builder e2 in
| let pointer = L.build_in_bounds_gep s_list [| L.const_int i32_t 0; L.const_int i32_t 0 |] "length" builder in
| let length = L.build_load pointer "size" builder in
| let list_pointer = L.build_in_bounds_gep s_list [| L.const_int i32_t 0; L.const_int i32_t 1 |] "cur_list_ptr" builder in
| let act_list = L.build_load list_pointer "cur_list" builder in
| let pointer_to_element = L.build_gep act_list [| index |] "pointer_to_element" builder in
| L.build_load pointer_to_element "tmp" builder

| A.ABlock(es, t) -> (match es with
| e:::e1:::rest -> ignore(expr builder e); expr builder (A.ABlock(e1:::rest, t))
| [e] -> expr builder e)
| A.APreop(op, e, _) ->
| let e' = expr builder e in
| (match op with
| A.Neg -> L.build_neg
| A.Not -> L.build_not
| ) e' "tmp" builder

(*
| A.AMuPreop(op, e, _) ->
(* given pointer to pitch array, memory operations for
adding or subtracting to position 0 of pitch element *)
(* index is needed so map will accept it *)
let interior_operation index pitch builder1 =
let prefield_pointer=L.build_gep pitch [| (L.const_int i32_t 0)|] "prefield_elem" builder1 in
let cur_prefield = L.build_load prefield_pointer
"cur_prefield" builder1 in
(match op with
A.AOup ->
let new_prefield = L.build_add cur_prefield
let new_prefield = L.build_sub cur_prefield
(L.const_int i32_t 1) builder1

ignore(L.build_store new_prefield preffield_pointer builder); in

(* match different things mupreops could be applied to, there are 3 *)
(match e with
  | A.APitch ->
    let e' = expr builder e in
    interior_operation e' builder
  | A.AChord ->
    let e' = expr builder e in
    map (get_list e' builder) interior_operation
  | A.AChordlist ->
    let e' = expr builder e in
)

(* A.AAssign (s, e, t) -> let e' = expr builder e in
  let var = try Hashtbl.find main_vars s
  with Not_found ->
    let local_var = L.build_alloca (ltype_of_typ t) s builder
    Hashtbl.add main_vars s local_var;
    local_var in
  ignore (L.build_store e' var builder); e'*

  | A.ACall (A.AID("Map", _), act, _) ->
    let func = expr builder (List.hd act) in
    let lst = expr builder (List.hd (List.tl act)) in
    let wrapper f = f in
    map lst (wrapper func); L.const_int i32_t 1
  *)

| A.ACall (A.AID("Printint", _), [e], _) ->
  L.build_call printf_func [] int_format_str ; (expr builder e) [] "printf" builder

| A.ACall (A.AID("Printstr", _), [e], _) ->
L.build_call printf_func [] str_format; (expr builder e) [] "printf" builder
| A.ACall (A.AID("Printfloat"),_), [e],_ ->
| L.build_call printf_func [] float_format; (expr builder e) [] "printf" builder
| A.ACall (A.AID("Printlist"),_), [e],_ ->
  let printfun value index builder = L.build_call printf_fun [] [int_no_line; value] "printf" builder in
  let s_list = expr builder e in
  map s_list printfun; L.const_int i32_t 1
| A.ACall(A.AID("Printlist"),_), [e],_ ->
  let printfun value index builder = L.build_call printf_fun [] [float_no_line; value] "printf" builder in
  let s_list = expr builder e in
  map s_list printfun; L.const_int i32_t 1
| A.ACall(A.AID("Make_midi"),_), [e1; e2],_ ->
L.build_call make_midi_func [] (expr builder e1); (expr builder e2) [] "make_midi" builder
| A.ACall(A.AID("Merge"),_), [e1; e2],_ ->
  (* assumed order of acutals: pitchlist, rhythmlist, modelist, start note, channel num *)
| A.ACall(A.AID("Synth"),_), act, _ ->
  (*extract the actuals *)
  let clist = expr builder (List.hd act) in
  let clist_len = get_length (clist, builder) in
  let rlist = expr builder (List.hd (List.tl act)) in
let act_rlist = get_list(rlist, builder) in
let modelist = expr builder (List.hd (List.tl (List.tl act))) in
let act_modelist = get_list(modelist, builder) in
let mode_len = get_length(modelist, builder) in
let start_pitch = expr builder (List.hd (List.tl (List.tl (List.tl act)))) in
let channel = expr builder (List.hd(List.tl(List.tl(List.tl(List.tl act))))) in

(*build the nessesary structures to pass into c function -
plist as non-struct int***, list of chord lengths, return-arr *)

(*malloced structure that contains lengths of chords *)
let chord_lengths = L.build_array_malloc i32_t clist_len
"return_arr" builder in

(*malloced structure that normalized pitch array (no
octaves or accidentals) will be built into. This is passed
into C synth function *)
let clear_cl_list = L.build_array_malloc i32p_t
clist_len "return_arr" builder in

(*building non-struct chord : Note that this refers to
both the normal builder and the builder inside the while loop
(builder1)*)
let passed_cl_list = L.build_array_malloc i32pp_t clist_len
"norm_arr" builder in

(* for chord_lengths *)
let pointer_to_ret_elem = L.build_in_bounds_gep
passed_cl_list [] [ index | ] "cur_val" builder1 in
let new_elem_list = L.build_extractvalue value1 1
"stuff" builder1 in

let chord_len_pointer = L.build_in_bounds_gep
chord_lengths [] [ index | ] "len" builder1 in
let new Elem_len = L.build_extractvalue value1 0
"oldlen" builder1 in

let clear_list_pointer = L.build_in_bounds_gep
clear_cl_list [] [ index | ] "len" builder1 in
let new_clear_arr = L.build_array_malloc i32_t
"clear_cl_list_elem" builder1 in
ignore(L.build_store new_clear_arr
clear_list_pointer builder1);
ignore(L.build_store new_elem_len
cord_len_pointer builder1);
ignore(L.build_store new_elem_list pointer_to_ret_elem builder1);

  in

  map clist (* (get_list(clist, builder)) *) chord_func;

  (* build buffer *)
  let buff = L.build_array_malloc i8_t (L.const_int i32_t 1000) "synth-buffer" builder in
  (* call synth *)
  ignore(L.build_call synth_func [| (* int **)passed_cl_list; (* int *)clist_len;
  (* int **)chord_lengths; (* int *) start_pitch; (*
  int ** )act_modelist;
  (* int **)mode_len; (* double **)act_rlist; (* int
  ** )clear_cl_list; (* int *)channel; (* char **) buff |]
  "synth" builder);

  buff

  | A.ACall (A.AID(s, _), act, _) ->
  let (fdef, fdecl) = Hashtbl.find function_defs s in
  let actuals = List.rev (List.map (expr builder)
  (List.rev act)) in
  let result = (match fdecl with
  A.AFun(f, _, _, _) -> f
  | _ -> raise(Failure "second problem with call") ) in
  L.build_call fdef (Array.of_list actuals) result builder

  | A.AIf(e1, e2, e3, _) ->
  let bool_val = expr builder e1 in
  let cur_fun = L.block_parent (L.insertion_block builder) in
  let merge_bb = L.append_block context "merge"
  cur_fun in
  let then_bb = L.append_block context "then" cur_fun in
  ignore(expr (L.builder_at_end context then_bb) e2);
  ignore(L.build_br merge_bb (L.builder_at_end context
then_bb));

let else_bb = L.append_block context "else" cur_fun in
ignore(expr (L.builder_at_end context else_bb) e3);
ignore(L.build_br merge_bb (L.builder_at_end context else_bb));
ignore(L.build_cond_br bool_val then_bb else_bb builder);
L.position_at_end merge_bb builder;
L.const_int i32_t 1

| A.AFun(fid, arg_list, e, A.TFun(arg_types, ret_type)) ->

let formal_types = Array.of_list (List.map ltype_of_typ arg_types) in
let ftype = L.function_type (ltype_of_typ ret_type) formal_types in
let the_function = L.define_function fid ftype the_module in
Hashtbl1.add function_defs fid
    (the_function, A.AFun(fid, arg_list, e, A.TFun(arg_types, ret_type)));
let builder2 = L.builder_at_end context (L.entry_block the_function) in
let alloc_local (s, t, p) =
    L.set_value_name s p;
let local_var = L.build_alloca (ltype_of_typ t) s builder2 in
ignore(Hashtbl.add main_vars s local_var);
ignore(L.build_store p local_var builder2) in
    let rec iter3 (f, l1, l2, l3) =
        (match (l1, l2, l3) with
            (hd1::rest1, hd2::rest2, hd3::rest3) -> f(hd1, hd2, hd3); ignore(iter3(f, rest1, rest2, rest3))
            | (_, [], []) -> ()
            | _ -> print_endline "ERROR LINE 491";
        ) in
iter3 (alloc_local, arg_list, arg_types, (Array.to_list((L.params the_function))));

let ret_val = expr builder2 e in
```plaintext
L.build_ret ret_val builder2

| _ -> L.const_int i32_t 1

let exprbuilder builder e = ignore(expr builder e);
builder

let builder = List.fold_left exprbuilder builder (List.rev(exprs))

ignore (L.build_ret (L.const_int i32_t 0) builder);
the_module
```
open Ast
open Lib

module StringMap = Map.Make(String)
module StringSet = Set.Make(String)
type environment = typ StringMap.t

module StringSet = Set.Make(String)
type constraints = (typ * typ) list

let letter = ref (Char.code 'a');;
let new_type () = let c1 = !letter in
  incr letter; TType(Char.escaped (Char.chr c1));;

let kws = ["if"; "then"; "else"; "true"; "false"; "def"];;

let keywords =
  List.fold_left (fun set x -> StringSet.add x set) StringSet.empty kws;;

let rec annotate_expr exp env : (aexpr * environment) =
  match exp with
  | Unit -> AUnit(TUnit), env
  | Literal(n) -> ALiteral(n, TInt), env
  | FloatLit(n) -> AFloatLit(n, TFloat), env
  | BoolLit(n) -> ABoolLit(n, TBool), env
  | String(n) -> AString(n, TString), env
  | ID(n) -> if StringMap.mem n env then
    AID(n, StringMap.find n env), env
    else raise(Failure(n ^ " Not Found"))
  | Binop(e1, op, e2) ->
    let ae1, _ = annotate_expr e1 env
    and ae2, _ = annotate_expr e2 env
    and ntyp = new_type () in
    ABinop(ae1, op, ae2, ntyp), env
  | Preop(preop, e) ->
    let ae, _ = annotate_expr e env
    and ntyp = new_type () in
    APreop(preop, ae, ntyp), env
  | Postop(e, postop) ->
let ae, _ = annotate_expr e env
and ntyp = new_type () in
APostop(ae, postop, ntyp), env
| Assign(name, e) ->
  if StringMap.mem name env then
    raise (Failure "Reassignment")
  else if StringSet.mem name keywords then
    raise (Failure "Redefining keyword")
  else let ntyp = new_type () in
  let nenv = StringMap.add name ntyp env in
  AAssign(name, ae, ntyp), nenv
| List(e_list) ->
  let ae_list = List.map (fun e -> fst (annotate_expr e env)) e_list in
  AList(ae_list, TList(new_type ())), env
  | RList(e_list) ->
    let ae_list = List.map (fun e -> fst (annotate_expr e env)) e_list in
    AList(ae_list, TList(TFloat)), env
| Pitch(i1, e, i2) ->
  let ae, _ = annotate_expr e env in
  APitch(i1, ae, i2, TPitch), env
| Block(e_list) ->
  let ae_list, nenv = ListLabels.fold_left ~init: ([], env) e_list
    ~f: (fun (ae_list, env) e -> let ae, env =
      annotate_expr e env in (ae::ae_list, env))
    in ABlock(ae_list, new_type ()), nenv
  | Chord(e_list) ->
    let ae_list = List.map (fun e -> fst (annotate_expr e env)) e_list in
    AList(ae_list, TList(TPitch)), env
  | ChordList(e_list) ->
    let ae_list = List.map (fun e -> fst (annotate_expr e env)) e_list in
    AList(ae_list, TList(TList(TPitch))), env
  | Call(func, args) ->
    let a_func, _ = annotate_expr func env in
    let a_args = List.map (fun arg -> fst (annotate_expr arg env)) args in
    ACall(a_func, a_args, new_type ()), env
  | If(pred, e1, e2) ->
let apred, _ = annotate_expr pred env
and e1, _ = annotate_expr e1 env
and e2, _ = annotate_expr e2 env in
AIf(apred, e1, e2, new_type()), env
|
Fun(name, args, e) ->
if StringMap.mem name predefined
then raise (Failure "Cannot redefine stdlib function.");
let args_t = List.map (fun f -> new_type()) args
and ret_t = new_type()
in
let fun_t = TFun(args_t, ret_t) in
let a_args = List.combine args args_t in
let nenv = List.fold_left
(fun e (id, t) ->
 if StringMap.mem id e
 then raise (Failure "Variable already defined")
 else StringMap.add id t e) env a_args
in
if StringMap.mem name env then
raise (Failure "Redefining function")
else let nenv = StringMap.add name fun_t nenv in
let ae, _ = annotate_expr e nenv in
AFun(name, args, ae, fun_t), nenv
|
Subset(var, e) ->
let avar, _ = annotate_expr var env
and ae, _ = annotate_expr e env
and t = new_type() in
let typ = t in
ASubset(avar, ae, t), env
|
_ -> AUnit(TUnit), env

let type_of ae =
match ae with
| AUnit(t) -> t
| ALiteral(_,t) -> t
| AFloatLit(_,t) -> t
| AString(_,t) -> t
| ABoolLit(_,t) -> t
| AID(_,t) -> t
| ABinop(_,_,_,t) -> t
| APreop(_,_,_,t) -> t
| APostop(_,_,_,t) -> t
| AAssign(_,_,_,t) -> t
let rec collect_expr ae : constraints =
  match ae with
  | ALiteral(_) -> []
  | ABoolLit(_) -> []
  | AFloatLit(_) -> []
  | AString(_) -> []
  | AUnit(_) -> []
  | AID(_) -> []
  | ABinop(ae1, op, ae2, t) ->
    let t1 = type_of ae1
    and t2 = type_of ae2 in
    let con = match op with
      | Add | Sub | Mult | Div -> [(t1, TInt); (t2, TInt); (t, TInt)]
      | FAdd | FSub | FMult | FDiv -> [(t1, TFloat); (t2, TFloat); (t, TFloat)]
      | Neq | Equal | Greater | Less | Geq | Leq -> [(t1, t2); (t, TBool)]
      | And | Or -> [(t1, TBool); (t2, TBool); (t, TBool)]
    in
    (collect_expr ae1) @ (collect_expr ae2) @ con
  | AAssign(_, ae, t) -> (collect_expr ae) @ [(t, type_of ae)]
  | ABlock(ae_list, t) ->
    let ret = List.hd (List.rev ae_list) in
    let ret_t = type_of ret in
    (List.flatten (List.map collect_expr ae_list)) @ [(t, ret_t)]
  | AList(ae_list, t) ->
    let list_t = match t with
    | TList(s) -> s
let con = List.map (fun aexpr -> (list_t, type_of aexpr)) ae_list in
    (list.flatten (List.map collect_expr ae_list)) @ con
| APitch(i1, ae, i2, t) -> [(type_of ae, TInt)]
| AIf(pred, ae1, ae2, t) ->
    let pt = type_of pred and t1 = type_of ae1 and t2 = type_of ae2 in
    let con = [(pt, TBool); (t1, t2); (t, t1)] in
    (collect_expr pred) @ (collect_expr ae1) @ (collect_expr ae2) @ con
| AFun(_, _, ae, t) -> begin match t with
| TFun(_, ret_t) -> (collect_expr ae) @ [(type_of ae, ret_t)]
| _ -> raise (Failure "Unreachable state in Function literal") end
| ASubset(v, e, typ) ->
    let vt = (match v with
        | AID(_) -> type_of v
        | _ -> raise (Failure "Unreachable state in Subset") ) in
    let s = match vt with
        | TList(t) -> [(t, typ)]
        | TType(t) -> [(vt, TList(typ))] in
    (collect_expr e) @ [(type_of e, TInt)] @ s
| ACall(name, args, t) ->
    let fnt = (match name with
        | AID(_) -> type_of name
        | _ -> raise (Failure "Unreachable state in Call") ) in
    let s = match fnt with
        | TFun(args_t, ret_t) ->
    begin
    let l1 = List.length args and l2 = List.length args_t in
    if l1 <> l2
then raise (Failure "Mismatched argument count")
else let args_c = List.map2 (fun ft at -> (ft, type_of at)) args_t args in
    args_c @ [(t, ret_t)]
end
| TType(_) -> [(fnt, TFun(List.map type_of args, t))]
| _ -> print_endline (string_of_typ fnt); raise (Failure "Mismatched type")
in (collect_expr name) @ (List.flatten (List.map collect_expr args)) @ s
| e -> raise (Failure ("collect_expr can't handle your expr yet" ^ string_of_aexpr e))
;
let rec substitute u x t =
  match t with
  | TUnit | TInt | TBool | TFloat | TPitch | TString -> t
  | TType(c) -> if c = x then u else t
  | TFun(t1, t2) -> TFun(List.map (substitute u x) t1, substitute u x t2)
  | TList(t) -> TList(substitute u x t)
  ;;
let apply subs t =
  List.fold_right (fun (x, u) t -> substitute u x t) subs t
  ;;
let rec unify constraints =
  match constraints with
  | [] -> []
  | (x, y) :: tl ->
    let t2 = unify tl in
    let t1 = unify_one (apply t2 x) (apply t2 y) in
    t1 @ t2
  
and unify_one t1 t2 =
  match t1, t2 with
  | TInt, TInt | TBool, TBool | TString, TString
  | TUnit, TUnit | TFloat, TFloat | TPitch, TPitch -> []
  | TType(x), z | z, TType(x) -> [(x, z)] (* Not completely correct *)
```ocaml
let rec apply_expr subs ae = match ae with
  | AUnit(t) -> AUnit(apply subs t)
  | ALiteral(value, t) -> ALiteral(value, apply subs t)
  | AFloatLit(value, t) -> AFloatLit(value, apply subs t)
  | ABoolLit(value, t) -> ABoolLit(value, apply subs t)
  | AString(value, t) -> AString(value, apply subs t)
  | ABinop(ae1, op, ae2, t) ->
      ABinop(apply_expr subs ae1, op, apply_expr subs ae2, apply subs t)
  | AID(name, t) -> AID(name, apply subs t)
  | APitch(a, ae, b, t) -> APitch(a, apply_expr subs ae, b, apply subs t)
  | AAssgn(name, ae, t) ->
      AAssgn(name, apply_expr subs ae, apply subs t)
  | AList(ae_list, t) ->
      AList(List.map (apply_expr subs) ae_list, apply subs t)
  | AFun(name, frmls, ae, t) ->
      AFun(name, frmls, apply_expr subs ae, apply subs t)
  | AIf(pred, ae1, ae2, t) ->
      AIf(apply_expr subs pred, apply_expr subs ae1, apply_expr subs ae2, apply subs t)
  | ACall(fname, args, t) ->
      ACall(apply_expr subs fname, List.map (apply_expr subs) args, apply subs t)
  | ABlock(ae_list, t) ->
      ABlock(List.map (apply_expr subs) ae_list, apply subs t)
  | ASubset(var, i, t) ->
      ASubset(apply_expr subs var, apply_expr subs i, apply subs t)
  | _ -> raise (Failure ("No apply_expr for AEXPR:" ^ string_of_aexpr e))
```

let infer expr env flag =
  let aexpr, nenv = annotate_expr expr env in
  let constraints =
    if flag then
      print_endline ("AEXPR: " ^ string_of_aexpr aexpr);
      collect_expr aexpr in
    let subs =
      List.iter (fun (a,b) ->
        if flag then
          print_endline
            ("CONSTRAINTS: " ^ string_of_typ a ^ " " ^
              string_of_typ b)) constraints;
        unify constraints in
      let inferred_expr =
        List.iter (fun (a,b) -> if flag then
          print_endline ("SUBS: " ^ a ^ " " ^ string_of_typ
            b)) subs;
            apply_expr subs aexpr in
        if flag then
          inferred_endline("FINAL: " ^ string_of_aexpr
            inferred_expr);
        inferred_expr, nenv
    ;;

let typecheck program flag : (aexpr list) =
  let env = Lib.predefined in
  let inferred_program, _ = ListLabels.fold_left (List.rev program)
  ~init: ([], env)
  ~f: (fun (acc, env) expr ->
    let inferred_expr, env = infer expr env flag in
    let inferred_expr, env = match inferred_expr with
      | AAssign(name, _, t) ->
        let env = StringMap.add name t env in
        inferred_expr, env
      | AFun(name, _, _, t) ->
        let env = StringMap.add name t env in
        inferred_expr, env
      | _ -> inferred_expr, env in

let infer expr env flag =
  let aexpr, nenv = annotate_expr expr env in
  let constraints =
    if flag then
      print_endline ("AEXPR: " ^ string_of_aexpr aexpr);
      collect_expr aexpr in
    let subs =
      List.iter (fun (a,b) ->
        if flag then
          print_endline
            ("CONSTRAINTS: " ^ string_of_typ a ^ " " ^
              string_of_typ b)) constraints;
        unify constraints in
      let inferred_expr =
        List.iter (fun (a,b) -> if flag then
          print_endline ("SUBS: " ^ a ^ " " ^ string_of_typ
            b)) subs;
            apply_expr subs aexpr in
        if flag then
          inferred_endline("FINAL: " ^ string_of_aexpr
            inferred_expr);
        inferred_expr, nenv
    ;;
(inferred_expr :: acc, env)
}

in (* List.rev *) inferred_program

;;
open Ast

module StringMap = Map.Make(String)

let stdlib = [
  ("Printint", TFun([TInt], TString));
  ("Printstr", TFun([TString], TString));
  ("Printfloat", TFun([TFloat], TString));
  ("Printlist", TFun([TList(TInt)], TString));
  ("Printlist", TFun([TList(TFloat)], TString));
  ("Synth", TFun([TList(TList(TPitch)); TList(TFloat); TList(TInt); TInt; TInt], TString));
  ("Make_midi", TFun([TString; TString], TUnit));
  ("Merge", TFun([TString; TString], TString));
];

let predefined =
List.fold_left (fun env (id, t) -> StringMap.add id t env)
StringMap.empty stdlib
;
#include <unistd.h>

int make_midi(char * buffer, char * name){
  execl("./testCFugueLib", "/testCFugueLib", buffer, name, 
  (char *)0);
  return 0;
}

# Make sure ocamlbuild can find opam-managed packages: first run
#
# eval `opam config env`
#
# Easiest way to build: using ocamlbuild, which in turn uses ocamlfind

all : musicmike.native synth.o make_midi.o

musicmike.native :
  ocamlbuild -ocamlyacc "ocamlyacc -v" -use-ocamlfind -pkgs
  llvm,llvm.analysis -cflags -w,+a-4 \
  musicmike.native

# "make clean" removes all generated files

.PHONY : clean
clean :
  ocamlbuild -no-log -clean
  rm -rf testall.log *.diff musicmike scanner.ml parser.ml
  parser.mli
  rm -rf synth make_midi
  *.output

parser:
  ocamlyacc -v parser.mly

scanner:
  ocamllex scanner.mll

frontend:
  ocamllex scanner.mll
  ocamlc -c ast.ml
  ocamlyacc -v parser.mly
  ocamlc -c parser.mli
  ocamlc -c lib.ml

# More detailed: build using ocamlc/ocamlopt + ocamlfind to
locate LLVM

37 OBJS = ast.cmx codegen.cmx infer.cmx lib.cmx parser.cmx
  scanner.cmx semant.cmx musicmike.cmx
39
40 musicmike : $(OBJS)
41    ocamlfind ocamlopt -linkpkg -package llvm -package llvm.analysis $(OBJS) -o musicmike
42
43 scanner.ml : scanner.mll
44    ocamllex scanner.mll
45
46 parser.ml parser.mli : parser.mly
47    ocamlyacc parser.mly
48
49 %.cmo : %.ml
50    ocamlc -c $<
51
52 %.cmi : %.mli
53    ocamlc -c $<
54
55 %.cmx : %.ml
56    ocamlfind ocamlopt -c -package llvm $<
57
58 # Synth from microC
59 synth : synth.c
60    cc -o synth -DBUILD_TEST synth.c
61 make_midi : make_midi.c
62    cc -o make_midi -DBUILD_TEST make_midi.c
63 ### Generated by "ocamldep *.ml *.mli" after building scanner.ml and parser.ml
64 ast.cmo :
65 ast.cmx :
66 codegen.cmo : ast.cmo
67 codegen.cmx : ast.cmx
68 musicmike.cmo : semant.cmo scanner.cmo parser.cmi codegen.cmo ast.cmo infer.cmo
69 musicmike.cmx : semant.cmx scanner.cmx parser.cmx codegen.cmx ast.cmx infer.cmx
70 parser.cmo : ast.cmo parser.cmi
71 parser.cmx : ast.cmx parser.cmi
72 scanner.cmo : parser.cmi
73 scanner.cmx : parser.cmx
semant.cmo : ast.cmo
semant.cmx : ast.cmx
infer.cmo : ast.cmo lib.cmo
infer.cmx : ast.cmx lib.cmx
lib.cmo: ast.cmo
lib.cmx: ast.cmx
parser.cmi : ast.cmo
(* Top-level of the MusicMike compiler: scan & parse the input, 
check the resulting AST, generate LLVM IR, and dump the 
module *)

type action = Ast | LLVM_IR | Sast | Compile | Semant

let _ = 
let action = if Array.length Sys.argv > 1 then 
List.assoc Sys.argv.(1) [("-t", Sast);(* Print the AST 
only *)
("-a", Ast);
("-l", LLVM_IR); (* Generate LLVM, don't 
check *)
("-c", Compile);
("-s", Semant)] (* Generate, check LLVM IR *)
else Compile in 
let lexbuf = Lexing.from_channel stdin in 
let ast = Parser.program Scanner.next_token lexbuf in 
let sast = 
match action with 
| Ast -> []
| Sast -> Infer.typecheck ast true
| Semant -> Semant.check (Infer.typecheck ast true)
| _ -> Semant.check (Infer.typecheck ast false) in 

match action with 
| Ast -> print_string (Ast.string_of_program ast)
| Sast -> print_string (Ast.string_of_inferred sast)
| LLVM_IR -> print_string (Llvm.string_of_llmodule
(Codegen.translate sast))
| Semant -> print_string ("SEMANT DEBUGGING :") ^
Ast.string_of_inferred (List.rev sast))
| Compile -> let m = Codegen.translate sast in 
Llvm_analysis.assert_valid_module m;
| Compile -> print_string (Llvm.string_of_llmodule m)
/* A program consists of a list of statements, aka `stmts` */

program: 

/* "stmts is a tuple with the first field being a list of
expressions (expr),
the second field being a list of function declarations
(fdecl), and the
third field being a list of type declaration (tdecl)" */

stmts:
  { [] }
  | stmts expr SEMI { $2 :: $1 }

/* "A function declaration `fdecl` consists of
a keyword 'Def'
a Function Identifier `FID` - string w/ first letter
capitalized
a list of formals `formals_list`
a body which consists of an `expr` expression "*/

fdecl:
  DEF FID formals_list ASSIGN expr { Fun($2, List.rev($3), $5) }

/* "expressions always return a value and consists of:
literals-basic types
binop-binary operator
unop-unary operators
primaries-miscellaneous pool (list type, assignment, etc.
"*/

expr:
  literals { $1 }
  | binop { $1 }
  | unop { $1 }
  | primaries { $1 }
  | fdecl { $1 }
  | LPAREN expr RPAREN { $2 }

literals:
LITERAL { Literal($1) }
| FLITERAL { FloatLit($1) }
| TRUE { BoolLit(true) }
| FALSE { BoolLit(false) }
| LPAREN RPAREN { Unit }
| ID { ID($1) }
| STRING { String($1) }

binop:
| expr PLUS expr { Binop($1, Add, $3) }
| expr MINUS expr { Binop($1, Sub, $3) }
| expr TIMES expr { Binop($1, Mult, $3) }
| expr DIVIDE expr { Binop($1, Div, $3) }
| expr FPLUS expr { Binop($1, FAdd, $3) }
| expr FMINUS expr { Binop($1, FSub, $3) }
| expr FTIMES expr { Binop($1, FMult, $3) }
| expr FDIVIDE expr { Binop($1, FDiv, $3) }
| expr EQ expr { Binop($1, Equal, $3) }
| expr NEQ expr { Binop($1, Neq, $3) }
| expr LT expr { Binop($1, Less, $3) }
| expr LEQ expr { Binop($1, Leq, $3) }
| expr GT expr { Binop($1, Greater, $3) }
| expr GEQ expr { Binop($1, Geq, $3) }
| expr AND expr { Binop($1, And, $3) }
| expr OR expr { Binop($1, Or, $3) }

unop:
| /*| MINUS expr %prec NEG { Preop(Neg, $2) } */
| NOT expr { Preop(Not, $2) }

primaries:
/* "Block of expressions" */
| LBRACE semi_list RBRACE { Block($2) }
/* "Calling a function "*/
| FID LPAREN actuals_list RPAREN { Call(ID($1), List.rev($3)) }
/* "Assigning a value to an variable"*/
| assign { $1 }
/* "list of expressions of same type (enforced in
semant.ml) */
123   | LBRACKET expr_list RBRACKET { List(List.rev($2)) }  
124     /* "list of chords" */
125   | PLBRACKET pxpr_list RBRACKET { ChordList(List.rev($2)) }  
126     /* "list of rhythms" */
127   | RLBRACKET expr_list RBRACKET { RList(List.rev($2)) }  
128     /* "tuple of expressions with different types (enforced in semant.ml)" */
129   /* | LTUPLE expr_list RTUPLE { Tuple($2) } */
130     /* "concatanating 2 lists (enforced in semant.ml)" */
131   | expr CONCAT expr { Concat($1, $3) }  
132     /* "If, then else" */
133   | IF expr THEN expr ELSE expr 
134       %prec IF  
135     { If($2, $4, $6) }  
136     /* "getting an element from a list/tuple/pitchlist" */
137   | ID DOTLBRACKET expr RBRACKET { Subset(ID($1), $3) }  
138 139
140     /* "Assigning a value to an variable"*/
141 assign:  
142     ID ASSIGN expr { Assign($1, $3) }  
143     /*" List of assingments (a=b) used in type declaration " */
144 assign_list:  
145     assign { [$1] }  
146     | assign_list assign { $2 :: $1 }  
147 148
149     /* "List of whitespace separated expressions used in 
150      -Lists 
151      -Tuples" */
152 expr_list:  
153     /*nothing*/ { [] }  
154     | expr_list expr { $2 :: $1 }  
155 156
157     /* "List of semicolon separated expressions used in block" */
158 159
160 161
162 163
164 semi_list:
*/ List of formal arguments used in function declaration */

formals_list:
  /*nothing*/               { [] }  
  | forms_list ID           { $2 :: $1 } 

  */ List of actual arguments used in function calls */

  actuals_list:
  /*nothing*/               { [] }  
  | actuals_list expr       { $2 :: $1 } 

  */ List of whitespace separated chords(simultaneous pitches) used in Plist (pitch list) */

pxpr_list:
  chord                     { [Chord($1)] } 
  | pxpr_list chord          { Chord($2) :: $1 } 

  */ List of simultaneous pitches */

  chord:
  pitch                     { [$1] } 
  | chord BAR pitch          { $3 :: $1 } 

  /* Tuple consisting of 3 fields: 
  prefield-a list of ints representing '^' and 'v' as '1' and '-1'
  an int representing scale degree inputed by user
  postfield-a list of ints representing '#' and 'b' as '1' 

  */p:[3|5|6  3 ^3#|9bb]*/
and '‐1' "*/

207    208    pitch:  209          pref Field expr postfield { Pitch($1, $2, $3) }
210
211
212    /*"a list of ints representing '^' and 'v' as '1' and '-1'
213    respectively" */

214    pref Field:
215    /* nothing */ { 0 }
216          | pref Field OUP { $1+1 }
217          | pref Field ODOWN { $1-1 }
218
219
220    /* "a list of ints representing '#' and 'b' as '1' and '-1'
221    "* /

222    post Field:
223    /*nothing*/ { 0 }
224          | post Field OCTOTHORPE { $1+1 }
225          | post Field FLAT { $1-1 }
226
1 (* Semantic checking for the MicroC compiler *)
2 open Ast
3 open Infer
4 module StringMap = Map.Make(String)
5
6 (* Hack for polymorphism compilation: take each polymorphic function and check when it is called. *)
7 (* Create a new Function Aexpr for each time it is called, with the specific type of the actuals. *)
8
9 let check (aexprs: aexpr list) =
10  let is_poly ae = match ae with
11    | AFun(_,_,_,TFun(f_t, r_t)) ->
12       let poly t = match t with
13        | TType(_) -> true
14        | _ -> false
15       in list.exists poly f_t
16    | _ -> false
17  in
18  let poly = List.filter is_poly aexprs in
19  let getname ae = match ae with
20    | AFun(fn,_,_,_) -> fn
21    | _ -> raise (Failure "What the hell you doin")
22  in
23  let poly_fnames = List.map getname poly in
24  let rec is_call ae = match ae with
25    | [] -> []
26    | AAssign(_,ie,_)::rest -> (match ie with
27      | ACall(AID(fn,t), args, rt) -> let name = fn in
28        if List.mem name poly_fnames then
29          ACall(AID(fn,t),args, rt)::(is_call rest)
30        else (is_call rest)
31        | _ -> is_call rest)
32      | ACall(AID(fn,t), args, rt)::rest -> let name = fn in
33        if List.mem name poly_fnames then
34          ACall(AID(fn,t),args, rt)::(is_call rest)
35        else (is_call rest)
36        | x::rest -> is_call rest
37    | x::rest -> is_call rest
let polycalls = is_call aexprs in
let rec matching x lst =
  match lst
  with [] -> []
  | ACall(AID(x, t), a, b)::rest -> ACall(AID(x, t), a, b)::
    (matching x rest)
  | y::rest -> matching x rest
  in
let poly t = match t with
  | TType(_) -> true
  | _ -> false
  in
let rec iterAexprs alist =
  match alist with
  | [] -> []
  | AFun(fn, a, b, TFun(ts, ret_t))::rest ->
    if List.exists poly ts then
      let callmatches = matching fn polycalls in
      let typelist cm =
        (match cm with
         | ACall(_, c, _) -> List.map Infer.type_of c
         | _ -> [])
        in
      let calltofun cm1 = AFun(fn, a, b, TFun(typelist cm1, Infe
```c
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
#include <unistd.h>
#include <errno.h>

int ***fold_lists ( int ***chord_list, int cl_length, int chord_lengths[], int start_pitch, int *modelist, int mode_length)
{
    //fprintf(stderr, "%s\n", "entering chord list");
    //map the mode to absolute pitches (0 corresponds to first scale degree)
int i=0;
    while (i<mode_length){
        modelist[i]=modelist[i]+start_pitch;
        //fprintf(stderr, "%s %d\n", "new mode value = ",
        modelist[i]);
        i+=1;
    }
    //runs off assumption that malloced chunks are not contiguous
int j=0;
    while (j<cl_length){
        int ** chord= chord_list[j];
        int i=0;
        //fprintf(stderr, "%s %d\n", "chord number", j);
        while (i<chord_lengths[j]){ //fprintf(stderr, "%s %d\n", "into while loop", i);
            //fprintf(stderr, "%p\n", chord);
            int * pitch= chord[i];
            //fprintf(stderr, "%s %d\n", "pitch number", i);
            //fprintf(stderr, "%s", "int *pitch=chord[i];");
            //new pitch to be added
            int transformed_pitch;
            int p=pitch[1];
            if (p==0){
                transformed_pitch=0;
            }
        }
    }
}
```c

else{
    int oup=(p-1)/mode_length;
    if (oup>0){
        //add 'overflow' to octaveup
        pitch[0]+=oup;
    }  
    transformed_pitch=modelist[(p-1)%mode_length];
    pitch[1]=transformed_pitch;
    i++;  
}

j++;  
}  
//fprintf(stderr,"%s\n", "after while loop");
return chord_list;
}

//takes normalized chord list and spits out list with actual pitches
int ** apply_accidentals (int ***chordlist, int cl_len, int *chord_lengths, int mode_length, int **return_arr, int * mode) 
{
    int octave=mode[mode_length-1]-mode[0];
    int j=0;
    while (j<cl_len){
        int ** chord= chordlist[j];
        int i=0;
        while (i<chord_lengths[j]){  
            int * pitch= chord[i];
            if (pitch[1]!=0){
                //add or subtract octaves
                int octave_shift=pitch[0]*octave;
                //add or subtract accidentals
            }  
            (return_arr[j])[i]=pitch[1];  
            i++;  
        }
    }
    return return_arr;
}  
```
j++;  
}  
return return_arr;  
}  

// string generator, takes rhythm list plus absolute pitch list and turns into strings that can be plopped in Cfugue  
// gonna mix in some c++ lets see if it crashes :/

int strgen (char * buff, double * rhythmlist, int ** corrected_chordlist, int cl_len, int * chord_lengths, int channel){  
    // first add channel to beginning
    char v[3];
    strcpy(v, "V");
    strcat(buff, v);
    char channel_buff[3];
    snprintf(channel_buff, 3, "%d", channel);
    strcat(buff, channel_buff);
    char space[2];
    strcpy(space, " ");
    strcat(buff, space);
    int j=0;
    while (j<cl_len){  
        // take note_len and convert into string
        double note_len= rhythmlist[j];
        // fprintf(stderr, "note_len: %.2f\n", note_len);
        char snote_len[10];
        memset(snote_len, '\0', sizeof(snote_len));
        snprintf(snote_len, 10, "%.2f", note_len);
        // initialize chord
        int * chord= corrected_chordlist[j];
        int i=0;
        while (i<chord_lengths[j]){  
            int pitch= chord[i];
            // convert pitch into string
            char pitchstring[3];
            snprintf(pitchstring, 3, "%d", pitch);
            // buffer all the goddamn symbols
            char lbracket[2];
strcpy(lbracket, "[");
char rbracket[3];
strcpy(rbracket, "]/");
char plus[2];
strcpy(plus, "+ ");
char space[2];
strcpy(space, " ");
char rest[3];
strcpy(rest, "R/");
strcpy(rest, ";");
strcpy(rest, "R/");
strcpy(rest, ";");

//make actual pitchstring
if (pitch==0){
    strcat(buff, rest);
}
else{
    strcat(buff, lbracket);
    strcat(buff, pitchstring);
    strcat(buff, rbracket);
}
    strcat(buff, snote_len);

if (i<chord_lengths[j]-1){
    strcat(buff, plus);
}
else{
    strcat(buff, space);
}
i++;
}
}
return 0;
}

//synth- imitates behavior of main(), compared at end

int synth(int *** chordlist, int len_chordlist, int * chord_lengths,
```c
int start_pitch, int * modelist, int mode_length, double *rhythmlist,
int **pure_chord_arr, int channel, char * buff)
{

   .fullName
    "%s\n", "in synth";
    };

    int *** new_chordlist = (int **) malloc(len_chordlist * sizeof(int **));

    int j1=0;
    while (j1<len_chordlist){
        fullName(\"%d\n", j1);
        fullName(\"%s\n", "LINE 162");
        int ** chord= chordlist[j1]; //old
        int ** new_chord = (int **) malloc(chord_lengths[j1]*sizeof(int *));
        new_chordlist[j1]=new_chord;//stuff in
        int i=0;
        while (i<chord_lengths[j1]){ //Line 167"
            fullName(\"%s\n", "LINE 167");
            int *pitch= chord[i]; //old
            int *new_pitch = (int *) malloc(3*sizeof(int));
            new_chord[i]=new_pitch;
            new_pitch[0] = pitch[0];
            new_pitch[1] = pitch[1];
            fullName(\"%s\n", "LINE 173");
            fullName(\"%s\n", "LINE 175");
            i++;
        }
        j1++;
    }

    int *new_modelist = (int *) malloc(mode_length * sizeof(int *));

    int j2 = 0;
    while(j2 < mode_length) {
        new_modelist[j2] = modelist[j2];
        j2++;
    }

    // int i = 0;
    // while (i<len_chordlist){
```
int **temp = (int **) malloc(chord_lengths[i]*sizeof(int *));

fprintf(stderr,"%s\n", "fuck pointers");
new_chordlist[i] = temp;
fprintf(stderr,"%s\n", "pointers are cooelio");
int j = 0;
while (j < chord_lengths[i]) {
    fprintf(stderr,"%s\n", "r = ", r);
    int* temp2 = (int*) malloc(3*sizeof(int));
    temp[j] = temp2;
    // (chordlist[i])[j] = temp2;
    int* pitch = (chordlist[i])[j];
    pitch[1] = r;
    pitch[2] = c;
    r++;
    j++;
}
i++;
}

int j = 0;
while (j < len_chordlist)
    int ** chord = chordlist[j];
    fprintf(stderr,"%p\n", chord);
    int i = 0;
    while (i < chord_lengths[j]){
        int* pitch = chord[i];
        fprintf(stderr,\"\t%p\n", pitch);
        fprintf(stderr,\"\t\t%d\n", pitch[0]);
        fprintf(stderr,\"\t\t%d\n", pitch[1]);
        fprintf(stderr,\"\t\t%d\n", pitch[2]);
        i++;
    }
    j++;
}
// modifies chordlist so mode is normalize to absolute value of notes. If range goes above octave, adds to prefield
int ***new_list = fold_lists(new_chordlist, len_chordlist, chord_lengths, start_pitch, new_modelist, mode_length);
234   //fprintf(stderr, "%s\n", "AFTER NEW LIST");
235   int j = 0;
236   while (j < len_chordlist) {
237       int ** chord = new_list[j];
238       //fprintf(stderr, "\%p\n", chord);
239       int i = 0;
240       while (i < chord_lengths[j]) {
241           int * pitch = chord[i];
242           //fprintf(stderr, "\t\%p\n", pitch);
243           //fprintf(stderr, "\t\%d\n", pitch[0]);
244           //fprintf(stderr, "\t\%d\n", pitch[1]);
245           //fprintf(stderr, "\t\%d\n", pitch[2]);
246           i++;
247       }
248       j++;
249   }
250   // copies new_list into pure_chord_list to incorporate octaves and accidentals (yes, I know a new int ** is redundant but atm just want to see if works
251   //fprintf(stderr, "%s\n", "after new_list");
252   int ** correct_pitches = apply_accidentals(new_list, len_chordlist, chord_lengths, mode_length, pure_chord_arr, new_modelist);
253   //fprintf(stderr, "%d\n", chord_lengths[0]);
254   // takes rhythm list and turns into string that can be fed into CFugue
255       memset(buff, '\0', 900);
256       strgen (buff, rhythm_list, correct_pitches, len_chordlist, chord_lengths, channel);
257       fprintf(stderr, "buff %s\n", buff);
258   return 0;
259 }
260   // //tester
261   // int main();
262   //
263   //
264   //
265   //
266
```c
270
271 // variables we need
272 // int ***chordlist;
273 // int cl_len=4;
274 // int chord_lengths[4]={2,2,2,2};
275 // int start_pitch=10;
276 // int modelist[4]={1,3,5,7};
277 // int mode_length=4;
278 // double rhythmlist[]={1, 1.5, 0.25, 0.33};
279 // build chordlist
280 // chordlist=(int ***)malloc(4 * sizeof(int **));
281 // int r=0; // pitch literal value
282 // int c=-1; // accidental value
283 // int i=0; // number of chords
284 // while (i<4){
285     // int **temp=(int **)malloc(2*sizeof(int *));
286     // fprintf(stderr,"%s\n", "fuck pointers");
287     // chordlist[i]=temp;
288     // fprintf(stderr,"%s\n", "pointers are cooeilo");
289     // int j=0;
290     // while (j<2){
291         // fprintf(stderr,"%s %d\n", "r = ", r);
292         // int* temp=(int *) malloc(3*sizeof(int));
293         // (chordlist[i])[j]=temp;
294         // int* pitch=(chordlist[i])[j];
295         // pitch[1]=r;
296         // pitch[2]=c;
297         // r++;
298         // j++;
299     }
300     // i++;
301     // }
302 
303 
304 
305     // fprintf(stderr,"%s\n", "chord list was created");
306     //
307     // int temp1[2];
308     // int temp2[2];
309     // int temp3[2];
310     // int temp4[2];
311     // int *modarr[4]={ temp1, temp2, temp3, temp4};
```
// testing synth
synth(chordlist, cl_len, chord_lengths, start_pitch, modelist, mode_length, rhythmlist, modarr);

// let's see...
int ***new_list = fold_lists(chordlist, cl_len, chord_lengths, start_pitch, modelist, mode_length);

// print it!
int j=0;
while (j<cl_len){
  int **chord = new_list[j];
  fprintf(stderr, "%p\n", chord);
  int i=0;
  while (i<chord_lengths[j]){  
    int *pitch = chord[i];
    fprintf(stderr, "%t%p\n", pitch);
    fprintf(stderr, "%t%d\n", pitch[0]);
    fprintf(stderr, "%t%d\n", pitch[1]);
    fprintf(stderr, "%t%d\n", pitch[2]);
    i++;
  }
  j++;
}

// correct_pitches=apply_accidentals(new_list, cl_len, chord_lengths, mode_length, modarr);

int j=0;
while (j<cl_len){
  int *chord = correct_pitches[j];
  fprintf(stderr, "%p\n", chord);
  int i=0;
  while (i<chord_lengths[j]){  
    int pitch = chord[i];
    fprintf(stderr, "%t%d\n", pitch);
    i++;
  }
  j++;
}
351
352
353 // // testing adding rhythm to the whole shebang
354 //   char buff[14* 8+1];
355 //   buff[0]=\0;
356 //   strgen (buff, rhythmlist, correct_pitches, cl_len,
357 //     chord_lengths );
358 //   fprintf(stderr,"%s\n", buff);
359
360 // // testing aggregate synth funtion
361
362 // return 0;
363 // }