Rhythm

Final Report

John Sizemore	jcs2213
Cristopher Stauffer	cms2231
Lauren Stephanian	lms2221
Yuankai Huo	yh2532

Department of Computer Science
Columbia University in the City of New York
New York, NY 10027 USA

Table of Contents:

1. Introduction

The Rhythm language provides a programmatic way to easily compose music from solos to elaborate symphonies through a novel programming language. By modeling a musical score as a programming language, elements such as notes, chords, and tracks can be used to programmatically create, edit, and play musical compositions. Rhythm lets you define functions for generation of musical content or to alter existing musical content.

2. Language Tutorial

This section contains a brief introduction to the Rhythm language. It is intended as a tutorial on the language, and aims at getting a reader new to started as quickly as possible.

2.1 Hello World ("Row, Row, Row Your Boat")

"Row, Row, Row Your Boat" is an English language nursery rhyme, and a popular children's song, often sung as a round. It is employed as Hello World example for Rhythm,

```
getBaseNotes() {
   def row;
   def rowbase;
   rowbase = [[C5,E5,G5], [C5,E5,G5], [C5,E5,G5],D5.8, E5.8, E5.8, D5.8, E5.8,
F5.8, G5.2, C6, G5, E5, C5, G5.8, F5.8, E5.8, D5.8, [C5,E5,G5]];

  row = rowbase->3;
  return row;
}

track_1() {
   return getBaseNotes();
}

track_2() {
   return R.1->4 :: getBaseNotes() << 2;
}

track_3() {
   return R.1->2 :: getBaseNotes() << 1;
}
```

First, the "getBaseNotes" function defines basic music array and repeats it 3 times. Then it is used by three tracks. Track 1 returns and writes the basic array, track 2 and track 3 stretches the basic array and changes the octave.

2.2 Environment Setting

<u>The following are required to run Rhythm:</u>

- OCaml 4.00.1
- Linux/Cygwin with shell script support
- JDK 1.7

Then go to the “src” folder and compile Rhythm by simply typing “make”. In order to get rid of the generated files, type “Make clean”.

2.3 Compiling and Running

Save the code in the file row.ry, then compile it by typing:

```
rhythm -i < row.ry
```

This creates a .rym bytecode file output.rym. Then you could generate the midi file by running java command:

```
java -jar rym2MIDI  output.rym  row.midi [instrument] [time resolution]
```

A MIDI file will be generated upon completion of the previous step.

3. Language Manual

3.1. Program Structure

Rhythm provides a tremendous amount of flexibility, however it requires some basic conventions in its program structure. A program consists of the following:

- Zero or more global variables
- At least 1 **track** function (defined in track section)
- An optional **init** function that is evaluated before track compilation
- Any number of support functions

When a program is compiled, the output will consist of 1 compiled track for each track function defined.

3.2. Lexical Conventions

3.2.1 Identifiers (Names)

An identifier is a sequence of letters, underscores and digits, the first character of which is a letter [a–z] or [A–Z]. In identifiers, upper and lowercase letters are different.

3.2.2 Keywords

The following identifiers are reserved for use as keywords and may not be used otherwise. These represent identifiers for control structure, constants, as well as system level function calls that may not be redefined.

`if`
`loop`
`else`
`while`
`return`
`true`
`false`

3.2.3 Literals

Integers

Integers are represented by 1 or more digits.
- Maximum supported digit is 0 to +2^31
- Behavior when constructing an integer bigger or smaller than this number is not defined.

Note

Note literals represent the most common units of a song, and are represented using the following notation: BaseNote|Modifier|Octave

BaseNote(Required): A-G (uppercase)
Modifier (Optional): #/b
Octave (Required) = 0-10

Valid Examples:
`C#5, D2, Eb9`

Invalid Examples:
`C, D12, Ebb5`

Duration for notes are defaulted to a quarter note designation, but can be modified through adding the notation .[duration], where duration is [1,2,4,8,16].

In situations where there does not exist a natural sharp or flat, such as between E and F, E#5 would equal F5. Similarly, Cb5 would equal B5.

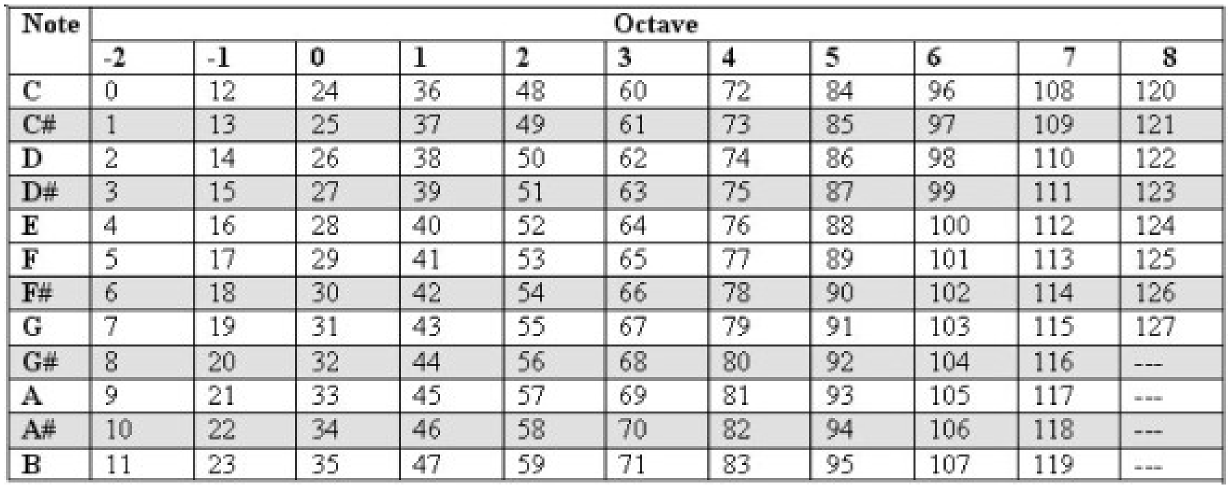
Rest

Rest literals represent a special kind of note, which has no pitch and therefore no volume. It is specified using the following symbol: R

Duration for rests are defaulted to a quarter note designation. Duration for rests are defaulted to a quarter note designation, but can be modified through adding the notation .[duration], where duration is [1,2,4,8,16].

Relationship of note/chord literals to octave and pitch

For simplicity in octave definition, this language defines the lowest octave as 0, which corresponds to -2 in the following table:



Note	Octave										
	-2	-1	0	1	2	3	4	5	6	7	8
C	0	12	24	36	48	60	72	84	96	108	120
C#	1	13	25	37	49	61	73	85	97	109	121
D	2	14	26	38	50	62	74	86	98	110	122
D#	3	15	27	39	51	63	75	87	99	111	123
E	4	16	28	40	52	64	76	88	100	112	124
F	5	17	29	41	53	65	77	89	101	113	125
F#	6	18	30	42	54	66	78	90	102	114	126
G	7	19	31	43	55	67	79	91	103	115	127
G#	8	20	32	44	56	68	80	92	104	116	---
A	9	21	33	45	57	69	81	93	105	117	---
A#	10	22	34	46	58	70	82	94	106	118	---
B	11	23	35	47	59	71	83	95	107	119	---

3.3. Arrays

Arrays represent a generic data structure capable of storing arbitrary sets of data, but for the purpose of this language, can be used to model measures, track segments, tracks, and songs.

3.3.1 Array Initialization

A new array can be defined through the use of brackets [ ] or through certain operators that concatenate literals together.

A basic array definition example is `[C5, C6, C7]`. Elements of the array are separated by commas.

An array may contain any combination of literals types, as well as other arrays. The following array is an entirely valid representation of a song: `[C5, C5 + 2, C5 + Eb6, [Gb6,A6,B#6], trackA ]`. In the previous example, the array contains a note, a note raised by one whole step, a chord, another chord, and a predefined track.

3.3.2 Array Access

An array can be accessed via the same bracket notation by including a non-negative integer value to denote the location with zero denoting the first location in the array.

Valid Example:

```
myArray[0], myArray[29], myArray[c], assuming c evaluates to an
integer
```

Invalid Example:
```
myArray[-5]
```

3.3.3 Array Modification
The same array access notation can be used to modify values as well. Values that replace the existing value can match in literal type or differ. However, an array cannot be enlarged by simply accessing cells beyond its initialized range. Combining and concatenating operations are covered in the expression section of this document.

Given:
```
myArray = [G2,E2,C2,B2,A2]
```

Valid Example:
```
myArray[0] = 99
```

Invalid Example:
```
myArray[99] = E2
```

3.3.4 Array Relationship to Notes, Chords, Tracks, and Songs
As stated in the beginning of this subsection, arrays are used to store collections of notes and chords through comma separated values indicating a preferred sequence. Ultimately, each song in Rhythm can be expressed as a list of lists; chords are simply a list of notes and tracks are a list of notes and chords. Songs by extension are then lists containing tracks, notes, and chords which, with the exception of notes, are themselves lists. This leads to a hierarchy of compositional levels within Rhythm songs as denoted below:

note -> chord -> track -> song

Subscripting of each compositional level will return entities of lower compositional levels but the reverse is not true.

3.4. Expressions
The below denote all expressions possible within Rhythm. The order of the major subsections indicate the precedence of those expressions; higher subsections will have higher precedence. All expressions will be applied from left to right.

3.4.1 Primary Expressions
The following denote primary expressions within Rhythm. These expressions are the fundamental types used to construct larger, more elaborate expressions.

identifier
Identifiers are lvalues denoting the name of an entity used within Rhythm. Identifiers can apply to functions, chords, tracks, or the songs themselves.

constant
Constants are either numeric literal integers, notes (A4), or rests (R).

expressions
Parenthetical expressions are used to establish arbitrary precedence within Rhythm programs.

identifier [numeric literal]
An identifier followed by a numeric literal in square brackets is used for subscripting purposes. Since songs can be represented by lists containing literals or other lists, this expression is used to access parts of a chord, track, or song.

identifier (empty or expression list)
This expression denotes a function call. The expression list within parentheses is used to pass in zero or more comma-separated arguments are per the specifications of the defined function.

[expression list]
This expression denotes a list of comma separated compositional types. This can be used to initialize a chord, track, or song value or can be used as a shorthand to anonymously declare chord, track, or song types within another chord, track, or song.

3.4.2 Modification Operators

expression + numeric literal
This expression takes all notes in the left operand and increases them all by the number of half steps specified by the right operand. Can also be used for mixing (see below)

expression - numeric literal
This expression behaves exactly the same as the previous expression except the notes in the left operand are decreased by the specified number of half steps.

expression++
This expression behaves as a shorthand method for quickly increasing the notes in the left operand by one half step.

expression--
This expression is exactly as the same as the previous one except all notes in the left operand are decreased by one half step.

expression * positive power of 2
The * operator denotes note duration lengthening. The full expression will take all the notes in the left operand and expand their pre-existing durations by the number specified by the right operand. Notes by default are assumed to be quarter notes. For example, if the left operand is a standard quarter note and the right operand is a 4, then the quarter note becomes a whole note. (e.g. A4 * 4 is a whole note)

expression / positive power of 2
The / operator also denotes note duration diminishing. For example, if the left operand is a standard quarter note and the right operand is a 4, the quarter note will become a sixteenth note. (e.g. A4 / 4 is a sixteenth note)

expression << numeric literal
This expression shifts all notes present in the left operand downward by the number of octaves denoted by the right operand.

expression >> numeric literal
This expression is the same as the one above except the notes contained in the left operand are shifted upwards by the specified number of octaves.

3.4.3 Combinational Operators

expression + expression + …
This expression denotes mixing one of or more operands meant to be played simultaneously. The result will be another expression equal in length to the longest operand.

expression :: expression :: …
This expression denotes concatenation of one or more expressions. The leftmost operand is considered to be the first entity to be played in the song followed by the operand immediately to its right and so on.

expression -> numeric literal …
This expression denotes stretching of an expression. The numeric literal indicates how many replicas are made. Useful for padding or making loops.

3.4.4 Equality Operators

expression == expression
This expression checks whether or not all notes within the two operands are equal to one another. Both operands must be of the same type.

expression != expression
This expression is exactly the same as the one above except the notes within the two operands are checked to see if they are unequal to one another. If even one note does not match then this expression will return true.

3.4.5 Assignment Operators

lvalue = expression
This expression takes an lvalue identifier as its left operand and assigns to it the expression denoted by the left operand.

lvalue += expression
lvalue -= expression
lvalue *= expression
lvalue /= expression
lvalue >>= expression
lvalue <<= expression
lvalue ::= expression

Expressions, in the form “exp1 =op exp2” is equivalent to the expression “exp1 = exp1 op exp2”. These expressions are used as a shorthand method for applying a particular operation to two operands and assigning the result to the lvalue specified by the left operand.

3.5. Statements

All statements below are considered to be sequential in nature. Statements are used to perform operations, assignments, function calls, or denote control flow.

3.5.1 Expression Statement

```
expression;
```

This statement, the most common of all statements in a Rhythm program, is used to denote an assignment, an operation, or a function call.

3.5.2 Conditional Statement

```
if (expression) {
      statement-list;
}
else {
      statement-list;
}
```

The if statement will take an expression as an argument and determine whether or not the expression is true. The argument must take the form of an equality expression denoted in Section 4.4. If this expression is true then the statement list enclosed by the braces immediately after the expression in parentheses will be executed. Otherwise, the statement list enclosed in braces following the else keyword will be executed. The else statement is optional.

3.5.3 While Statement

```
while (expression) {
      statement-list;
}
```

The `while` statement take an expression as an argument and executes the code within the statement list repeatedly until the condition specified by the argument is no longer true. The conditional expression is evaluated at the beginning of each cycle of execution and must take the form of an equality expression as shown in Section 4.4.

3.5.4 Loop Statement

```
loop (integer : integer) {
      statement-list;
}
```

The `loop` statement acts as a shorthand version of the familiar “for” loop seen in many other languages. Basically, the statement list is executed given the range of operations specified in the loop argument; if the LHS is 0 and the RHS is 6, then the loop will evaluate 6 times.

3.5.5 Return Statement

```
return;
return expression;
```

This statement is only used when terminating a function call. After performing the statements within a function, it will return to its caller with either no value or a specified value.

Special Note: Functions that begin with the name “track_” are required to return an expression value. This return value is used by the compiler to generate track data based on notes and chords returned. More details are provided in the section 6.

3.6. Functions

3.6.1 Defining A Function

Functions will be able to return values, but because the language contains no types, there is no return type in the function declaration. Function declarations consist of only the name of the function followed by parentheses between which are parameters separated by commas. After that, the function body will sit between two curly brackets.

```
concatMusic(track1, track2){
     /*concat track1 and track2*/
     /*return statement*/
}

concatMusic(intro, bridge);
```

3.6.2 Return Statement

Every function can contain a return statement, which is followed by the reserved word return. It would be unusual to declare a function that does not return any sort of musical value, but it is not necessary to always do so. It is possible for the function to return nothing.

3.6.3 Init Function

The init function is optional and is guaranteed to be evaluated before any track functions. The init function can be used to initialize global data that may be used by other functions. The init function should take no parameter:

```
      def globalValue;

      init() {
            ...initialize global value
      }
```

3.6.4 Track Functions

The compilation process relies on the definition of track functions. Tracks are defined identically to other functions, however the compilation process will compile and evaluate these functions individually, and then write out track data for each function. A track function must return either a note, rest, or array of notes and rests. If the function does not return a value, compilation will fail:

```
      track_myFirstTrack(){
            return [C5,D5,E5];
      }
```

3.7. Scope Rules

3.7.1 Variable Scope

Brackets will generally control the scope of variables. Variables declared inside brackets will not be accessible outside of those brackets because they will have local scope. This allows the user of this language to utilize the same identifying name multiple times as long as those variables are not within the same scope. Any variable declared outside of brackets will have global scope. If the name of a global variable is redefined inside of a set of brackets, it will overwrite the global definition of the variable only within the brackets.

```
bridge = [A5, D5, G5];
chorus;
introDrums;
introGuitar;
end;

concatMusic(){
      bridge = [] /*overwrites the global variable bridge declared
      above but only within this function. This bridge contains
      nothing*/
      song = (introDrums +  introGuitar) :: bridge :: chorus :: end;
      return song;
}
track_mysong()
```

```
{
      return concatMusic();
}
```

3.7.2 Function Scope

Function scope will be global. After a function has been declared, it may be called anywhere.

```
noteManipulation(myNote){
      /*manipulate notes*/
}

noteManipulation(A5);
harmonize(A5); /*will cause error - function has yet to be defined*/
```

3.8. Comments

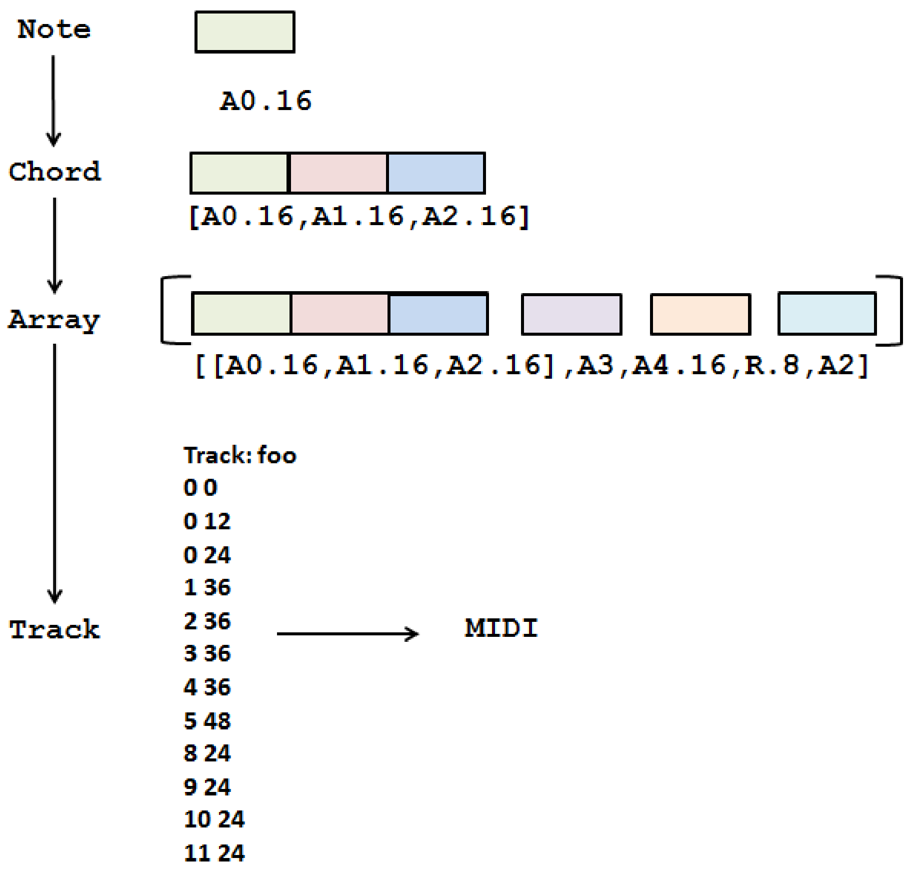
/* and */ will be used to indicate the beginning and end of a commented section of code. Anything written inside these characters will be ignored. These comments can be written on a single line or can span multiple lines.

```
/*This is a comment*/

/*This
is
another
comment*/
```

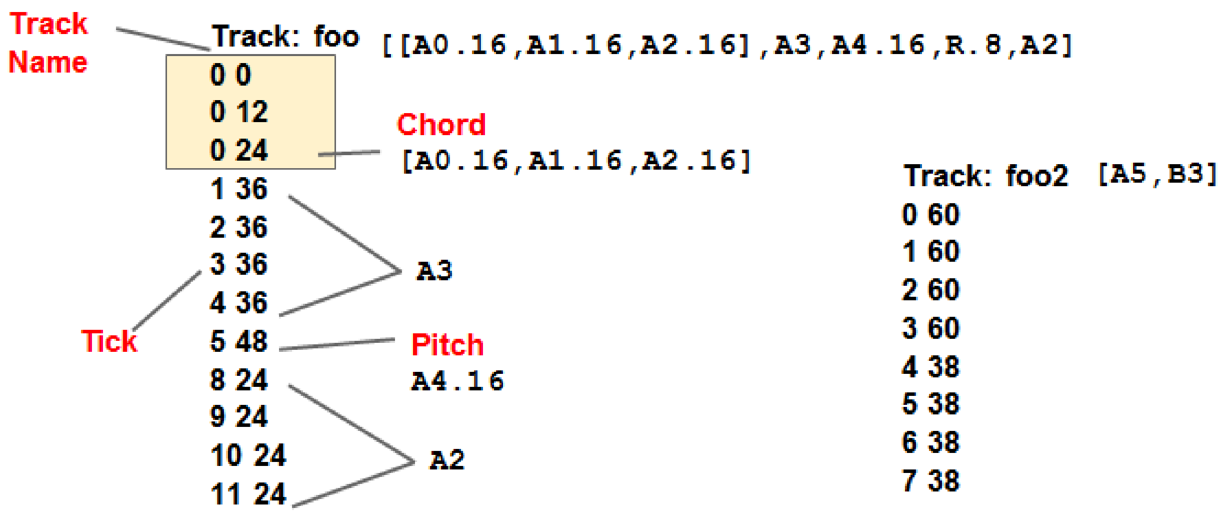
3.9. File Format and Output

Outputting a song in Rhythm follows the pipeline below:

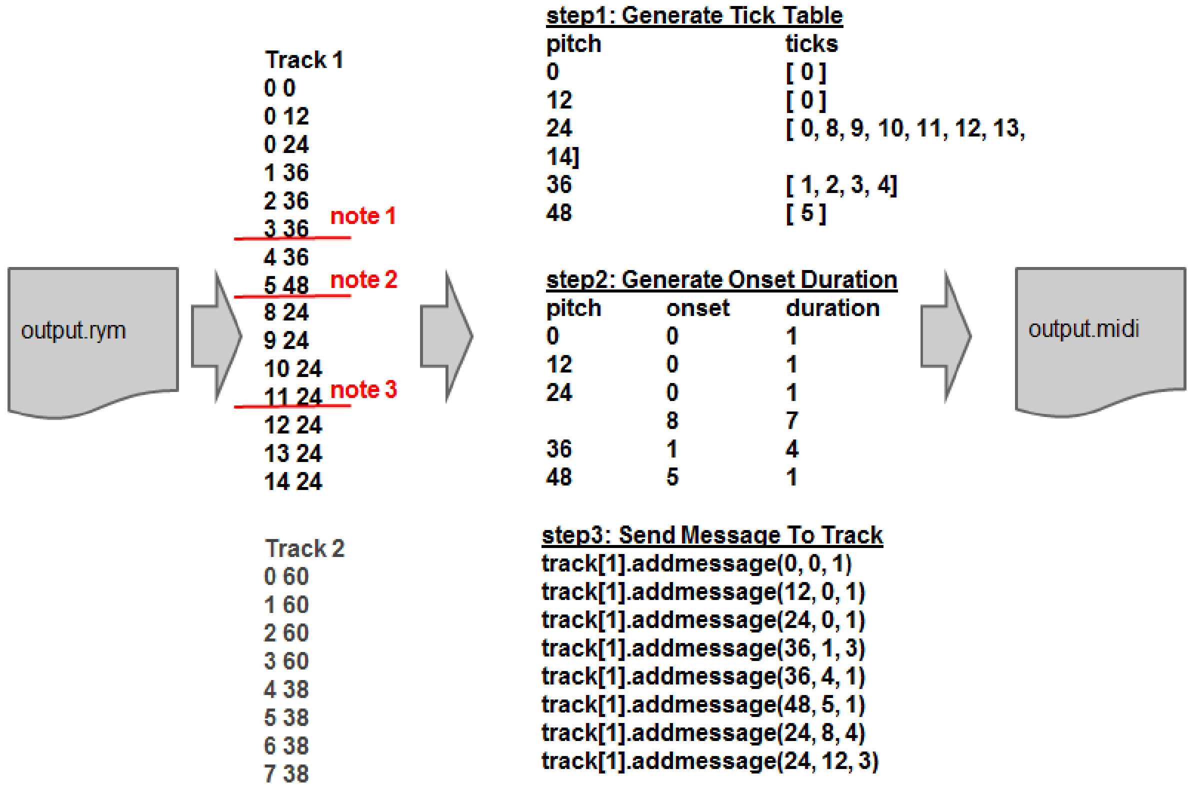
Note → Chord → Array → Track → MIDI



3.9.1 .ry file → .rym file



3.9.2 .rym file → midi file



```
Track 1
0 0
0 12
0 24
1 36
2 36
3 36
4 36
5 48
8 24
9 24
10 24
11 24
12 24
13 24
14 24

Track 2
0 60
1 60
2 60
3 60
4 38
5 38
6 38
7 38
```

step1: Generate Tick Table

pitch	ticks
0	[0]
12	[0]
24	[0, 8, 9, 10, 11, 12, 13, 14]
36	[1, 2, 3, 4]
48	[5]

step2: Generate Onset Duration

pitch	onset	duration
0	0	1
12	0	1
24	0	1
	8	7
36	1	4
48	5	1

step3: Send Message To Track

```
track[1].addmessage(0, 0, 1)
track[1].addmessage(12, 0, 1)
track[1].addmessage(24, 0, 1)
track[1].addmessage(36, 1, 3)
track[1].addmessage(36, 4, 1)
track[1].addmessage(48, 5, 1)
track[1].addmessage(24, 8, 4)
track[1].addmessage(24, 12, 3)
```

4. Project Plan

4.1 Process Overview

The Rhythm team used a **discovery** phase followed by an **iterative develop / test** phase.

The **discovery** phase of the process included:

- Braining-storming what an program that played the song “row row row your boat” might look like. Each person independently wrote their own program, and then we sat down and discussed the merits of each approach.
- Itemizing primitives and expression types that may be valuable.
- Itemizing the horizontal layers that needed to recognize, interpret, and compile these primitives and expressions
- Identifying any unique problems that may required separate research
    - Midi format
    - Note, Chord arithmetic

- The **setup** phase was responsible for setting up the tools for development: Version Control, make files, etc.

The **iterative develop/test** phase attempted to do the following:

- Identify vertical slices of development (e.g. primitives, functions, program, operators, expressions)
- Assign slice, and have an individual add all required code to all layers, and tests as well
- Run tests
- Commit and push changes to version control

4.2 Programming Style Guide

Our team did not adopt any format style guide for development. We all used the same editor (Sublime), which made for consistent formatting, however we adopted a “consistent with last code push” approach to writing code.

Generally, as Rhythm provides a modular interface for composition it is recommended to have programs follow this approach. Separating tracks into different pieces is also recommended as well as naming your variables and tracks something appropriate such as “bridge” or “intro” or “cMajor”. The operators can perform many of the same functionalities many other languages would require looping for; use these operators (e.g. mixing, stretching) to their fullest abilities. Lastly, it is recommended to use local variables as much as possible in order to minimize the usage of memory as well as to sidestep some potentially perplexing logical errors.

4.3 Roles and Responsibilities

Common: Primitive / Expression Definition, Example Authoring
John: Leader, Version Control, arrays, operators
Cris: Program Structure, functions, note/rest arithmetic
Lauren: File Writer
Kai: Control Structures, statements, rym & MIDI Format

4.4 Development Environment:

- Github Repository
- Ocaml binaries installed on desktop
- Sublime Text used for coding
- “Make” was used for the build process

4.5 Project Timeline and Log

July 10 - July 17 :

- Brainstorm programming language
- Setup Tools and Version Control System

July 18 - July 24:

- Itemize primitives, expressions, functions
- Define abstract syntax tree
- Research Midi format

July 24 - July 31:
- Write and test notes, rests, functions, global variables, operators
- Write and test primitive arithmetic module

August 1 - August 7:
- Write and test arrays and operators
- Write and test file rym file writer

August 8 - August 15:
- Write support for init and track_* methods for automated compilation
- Write and test file rym file writer
- Write example programs

```
commit 6aa7e6284a2e6025d9cedc72f0d2e382984fb07e
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 16 22:41:41 2013 -0400

    Final commit

commit 63edccb0dddda5c440fde197849ca7cfe41caaa7
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Fri Aug 16 22:32:11 2013 -0400

    adding examples

commit 1d8d29aed71219f5d71dbe3c5cfa953b4ec28389
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Fri Aug 16 20:43:36 2013 -0400

    Updating notes

commit 90b3bb11c079023c3df760d938db4d27d20e5e39
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Fri Aug 16 18:42:28 2013 -0400

    update

commit 6b4e9eba60386a5a6a9ed4a3bf8d7dacf17021bc
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Fri Aug 16 18:35:20 2013 -0400
```

```
    rm

commit 23c53b57aea4d0a3a30fe93ec6e865e20e35a656
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Fri Aug 16 16:48:39 2013 -0400

    one more

commit f6530692c70a9a1552a6745ef474a270067d499a
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Fri Aug 16 16:42:27 2013 -0400

    midi examples

commit ce676065bc6b2a213ca3605cdff738b5ad8f4492
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Fri Aug 16 15:55:25 2013 -0400

    midi examples

commit e32a0dbd7834f2b530a2a95ce3999750af0da34c
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Fri Aug 16 15:53:54 2013 -0400

    final version of rym2MIDI

commit 9b12879baed95e9babeeab553b3597b623c77aaa
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 16 04:11:14 2013 -0400

    Removed the teacher's test files I thought I got rid of

commit 246cb039864d006557d549ff064e9bee59fb0408
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 16 04:03:53 2013 -0400

    Consolidated files in their appropriate places

commit a3f663f5291f6c347516d6cb617465f558518683
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 16 04:02:47 2013 -0400
```

```
    Consolidated files in their appropriate places

commit b29bbfd57267ad80551c17a71d809b941f9c02ac
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 16 03:51:50 2013 -0400

    Got rid of teacher's test cases

commit c6ed4bec0f74e2fa8d9ca926fb05d0d564e7cfc6
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 16 03:50:56 2013 -0400

    Tweaked test cases to be more coherent

commit b7471bbcb9695592eff7d6904247ac67cb1da65a
Author: John <jcsizemo@gmail.com>
Date:   Thu Aug 15 18:04:08 2013 -0400

    Tweaked duration ops

commit d2dc143b10730f442372b84bf822efbce4d39d3b
Author: John <jcsizemo@gmail.com>
Date:   Thu Aug 15 17:57:14 2013 -0400

    Removed unused scanner tokens

commit dfd4fb3913e2915278716cd8dc1012c667f94f31
Author: John <jcsizemo@gmail.com>
Date:   Thu Aug 15 17:45:08 2013 -0400

    Added Gangnam Style track and folder

commit 49821e77a91cd44c84c405f40977a4389c04f5c5
Author: John <jcsizemo@gmail.com>
Date:   Thu Aug 15 17:09:34 2013 -0400

    Added row row row your boat

commit ff104bca5e4d638b346b6806e067031e85ea9cf4
Author: John <jcsizemo@gmail.com>
```

```
Date:   Thu Aug 15 16:16:03 2013 -0400

    Changed ++ to assign, cleaned a few things

commit c56bd6012177b26fb61bb0a8a3f23fe3e873101c
Author: John <jcsizemo@gmail.com>
Date:   Thu Aug 15 15:51:49 2013 -0400

    Cleaned up a few things

commit cf0e27584f23efe5d3132908636d77c1cabb701c
Author: John <jcsizemo@gmail.com>
Date:   Thu Aug 15 03:51:01 2013 -0400

    Fixed stretch operator

commit d7b6f60b9fae471fd29bcdbf97bc0e552f9b159e
Author: John <jcsizemo@gmail.com>
Date:   Thu Aug 15 03:48:55 2013 -0400

    Removed unnecessary files

commit 861409ff4ae8f494d8f4d12b8dac7654ddfce82e
Author: John <jcsizemo@gmail.com>
Date:   Thu Aug 15 00:39:14 2013 -0400

    Got rid of extra Shepard tone file

commit 816a1f3e8567efa1c3e2dc93792e084b72734d7e
Author: John <jcsizemo@gmail.com>
Date:   Wed Aug 14 23:28:56 2013 -0400

    Added stretch operator, got rid of warnings

commit de2530a1a5b71fdf6b818fd80a87bd8d026084a1
Author: John <jcsizemo@gmail.com>
Date:   Wed Aug 14 22:54:19 2013 -0400

    Added shepard tone example

commit 914cfcbdeb0c4fd03e60390d5ca5e3647463b74a
```

```
Author: John <jcsizemo@gmail.com>
Date:   Wed Aug 14 22:35:22 2013 -0400

    Changed output to start at zero

commit f4d6223e49d5147f77779247884879bcc6c4c15a
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Aug 13 11:42:00 2013 -0400

    update

commit ddcab29d2a60ba575439343f45af8bb5da069420
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Aug 13 11:39:17 2013 -0400

    update rym2MIDI.jar

commit 7e32de5826bbaa3995c2ae60b79f943c62aea1d5
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Tue Aug 13 09:00:29 2013 -0400

    Adding init function support

commit d30ba9b64ea18691a9c03e8ec4d1116c15c4f012
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Tue Aug 13 08:22:11 2013 -0400

    Changing logic to write out track functions only

commit 5ea5cb76cafa486f06ab94a0353357e11edea699
Author: John <jcsizemo@gmail.com>
Date:   Tue Aug 13 05:27:09 2013 -0400

    Added code and output for Shepard note illusion

commit a9dee3f505eebfb6357f81bea8dd048e682395c8
Author: John <jcsizemo@gmail.com>
Date:   Tue Aug 13 05:07:19 2013 -0400

    Added track info to RYM file, created MIDI file
```

```
commit b19b984eac4b08e3ca41c85a10f3a53c34e79e03
Author: John <jcsizemo@gmail.com>
Date:   Tue Aug 13 04:45:21 2013 -0400

    Modified rests to simply update tick count

commit a79b01d59080bf13bbd3f4d88f9c29aaad63305a
Author: John <jcsizemo@gmail.com>
Date:   Tue Aug 13 04:28:45 2013 -0400

    File output works with arrays

commit 45903fb28196234b3ff1756e568a101a22791420
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Aug 13 00:06:54 2013 -0400

    rym2ry

commit 1a006b9fb88a614f786b288ef4bece918e9cbd83
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Aug 13 00:04:56 2013 -0400

    rym2MIDI with instrument and time resolution

commit 6fe3fee9bf236594d869214d70a90a179f0c2559
Merge: 63d3119 95cad47
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Mon Aug 12 18:38:20 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

commit 63d3119c6475d62f6581042758f58d4828007b57
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Mon Aug 12 18:38:01 2013 -0400

    Adding duration write logic

commit 95cad47dde75490f5c949448e5ce04391ef3824c
Author: John <jcsizemo@gmail.com>
Date:   Sun Aug 11 21:24:56 2013 -0400
```

```
    Modification operators work on rests

commit 70046a36eed9670dde90ccf44f951240552cd354
Merge: 7f4c707 8702338
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sun Aug 11 19:04:39 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

    Conflicts:
      src/interpret.ml

commit 7f4c70764a0100d154c24f2f0592fd3d99b4a4bd
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sun Aug 11 18:52:22 2013 -0400

    Adding rests

commit 8702338c6f7cec352616591641862f61a5a7fbef
Author: Lauren Stephanian <lms2221@columbia.edu>
Date:   Sun Aug 11 11:27:33 2013 -0400

    Update interpret.ml

commit d0bef96a11334aaff6c1275c9ca1c81fecae26f0
Author: John <jcsizemo@gmail.com>
Date:   Sun Aug 11 04:36:47 2013 -0400

    Cleaned up operators

commit f7c5a372a582bdb5de9887acfc0ae022878dac57
Author: John <jcsizemo@gmail.com>
Date:   Sun Aug 11 04:23:23 2013 -0400

    Got +/-/>>/<</++/-- ops to work with notes with duration values

commit 3a785d3da0a1a6253e53c4bc02af0936d4336e5e
Author: John <jcsizemo@gmail.com>
Date:   Sun Aug 11 04:05:34 2013 -0400

    Fixed problem with array ops at the ends of arrays
```

```
commit bb1e03a23712bf30a67e9a2a3fb4cd47a6f0df42
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Aug 10 23:08:05 2013 -0400

    gangnam style

commit a11adef949c94f632d5fa698a7fdfc3e76db168b
Author: John <jcsizemo@gmail.com>
Date:   Sat Aug 10 16:26:47 2013 -0400

    Inc/Dec duration operation works on arrays

commit 064c640d69b9c06fd98e7689998a09d4460bd7fd
Author: John <jcsizemo@gmail.com>
Date:   Sat Aug 10 16:16:53 2013 -0400

    Got rid of unnecessary files, reduced Makefile

commit d27814d322f5fd8f5fdbc384d83274a482abf2d9
Author: John <jcsizemo@gmail.com>
Date:   Sat Aug 10 16:05:46 2013 -0400

    Got rid of unnecessary files

commit c444d76813bb68d8867366abfe39ae61dcb700f8
Merge: ef607c9 80deedd
Author: John <jcsizemo@gmail.com>
Date:   Sat Aug 10 16:03:15 2013 -0400

    Cleaned up all warnings

commit ef607c90a53d7c364bea13bd3037a66985cb6fc9
Author: John <jcsizemo@gmail.com>
Date:   Sat Aug 10 15:56:08 2013 -0400

    Got rid of all warnings

commit 80deedde7bfa008557987d1bc4c0fe110e285fdc
Merge: 9b61a63 3254cb2
Author: lms2221@columbia.edu <lms2221@columbia.edu>
```

```
Date:   Sat Aug 10 14:04:19 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

commit 3254cb218d45aab99b73be670fe9b44180a9774a
Author: John <jcsizemo@gmail.com>
Date:   Sat Aug 10 13:54:47 2013 -0400

    Finishing up operators: added oct up/down, half up/down, cleaned up
addition

commit 9b61a631d4f27a04f04aa92585d99942f836f94b
Author: lms2221@columbia.edu <lms2221@columbia.edu>
Date:   Thu Aug 8 13:48:17 2013 -0400

    added duration output

commit a04b213346906bdeacc103244cff91bbc428bd17
Author: lms2221@columbia.edu <lms2221@columbia.edu>
Date:   Thu Aug 8 13:38:45 2013 -0400

    added duration output

commit 4701172892be61bc7b7445ebcf68be426d9f6bd2
Author: lms2221@columbia.edu <lms2221@columbia.edu>
Date:   Thu Aug 8 13:37:01 2013 -0400

    added duration output

commit 77f5ffc505b41eaa2809bb133e75f86a41906058
Author: lms2221@columbia.edu <lms2221@columbia.edu>
Date:   Thu Aug 8 13:26:29 2013 -0400

    added duration output

commit 1e76459f1d80c6ce9db5c233641e74cf3e6639d5
Merge: d873545 4c264cc
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Wed Aug 7 18:41:38 2013 -0400

    Fixing conflicts
    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

    Conflicts:
      src/ast.ml
      src/datalib.ml
      src/interpret.ml
      src/parser.mly
      src/scanner.mll

commit d87354587fd4f8bce746b42e3fb7341c4b3ece38
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Wed Aug 7 18:27:46 2013 -0400

    Adding operators for duration

commit 4c264ccb9cada9c504f34c22759f339a26bb5e13
Author: Lauren Stephanian <lms2221@columbia.edu>
Date:   Wed Aug 7 17:35:53 2013 -0400

    Update interpret.ml

commit b093246bd33ff08c8eb082ad4ea3bcba92b8368e
Author: John <jcsizemo@gmail.com>
Date:   Wed Aug 7 01:35:33 2013 -0400

    Added op assign operators, cleaned up a few things

commit 6f5d9634e4e6300d3b5ea40f3d0ddba2e2c2e146
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Wed Aug 7 01:11:38 2013 -0400

    update

commit 30fd0e97878e601393f906ca4d1bfbd11a7ded09
Merge: cb3aca0 0dd41cf
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Wed Aug 7 01:10:23 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm
```

```
commit cb3aca090a0db11ad61abc405acac6db6532cc75
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Wed Aug 7 01:10:03 2013 -0400

    update

commit 94538ba7ae27645d70e636c5d752a7fb8ad23526
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Wed Aug 7 01:07:52 2013 -0400

    rym2MIDI_update2

commit 0dd41cf284cd2f5f14e44a572b3f7cdb3dbe1574
Author: John <jcsizemo@gmail.com>
Date:   Wed Aug 7 00:58:47 2013 -0400

    Added concat operator, cleaned up a few messages

commit dc28e636f3d139f7542c2ce7824d7d1135a97c8f
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Aug 6 22:54:18 2013 -0400

    add rym2MIDI

commit a60f325f0c981babfb01862cc539fd2cce6ce20f
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Aug 6 22:53:47 2013 -0400

    remove CSV2MIDI

commit 4a13a5132ab6b8fca553f8fc879fd43ae93e888f
Author: John <John@JohnDesktop2.(none)>
Date:   Tue Aug 6 18:06:08 2013 -0400

    Cleaned up mixing operation

commit ba2063df550d99a0334724376318d40d5183f6b5
Author: John <John@JohnDesktop2.(none)>
Date:   Tue Aug 6 16:28:06 2013 -0400

    Added array mixing
```

```
commit 7415521c84cc653529ee857f906c9223a3bbbd58
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Aug 6 00:26:40 2013 -0400

    update concat function

commit 37576b6dfbe31d721c255b85dbe7dd65b817d38c
Author: John <jcsizemo@gmail.com>
Date:   Mon Aug 5 02:59:30 2013 -0400

    Got note/literal addition working

commit 6e2873cad5b703b5e589946f594c62f9f4f590c2
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sun Aug 4 22:06:53 2013 -0400

    Adding modifier operators for notes

commit 5eaabf0565f5a0e03f6c6adf9a1990eb1b9fc1f6
Merge: 75e5450 82a3718
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sun Aug 4 21:06:55 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

    Conflicts:
      src/scanner.mll

commit 75e5450a2be840b2d4fbe0fca228d4adb61a7797
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sun Aug 4 21:03:48 2013 -0400

    Adding duration to note format

commit 82a3718f092106535754e9ef518032db9faff376
Author: John <jcsizemo@gmail.com>
Date:   Sun Aug 4 20:58:22 2013 -0400

    Got rid of shift/reduce conflicts in parser
```

```
commit 090bb3163be3ca8d61872315c1565c4d55d10a2e
Merge: 75494e1 012ceb4
Author: lms2221@columbia.edu <lms2221@columbia.edu>
Date:   Sun Aug 4 16:43:58 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

commit 75494e1cde8e0cc80e3a6c2cf37d1eb08ea87cad
Author: Lauren Stephanian <lms2221@columbia.edu>
Date:   Sun Aug 4 16:38:28 2013 -0400

    adding file write functionality

commit f892498e78f305c5c21a8d9553cce92d85487ac4
Author: Lauren Stephanian <lms2221@columbia.edu>
Date:   Sun Aug 4 16:37:13 2013 -0400

    adding file

commit 012ceb4f05b09de2a59dc7a588cccfb21a446b26
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sun Aug 4 16:26:00 2013 -0400

    implement for, while, loop statements

commit 06ddde97612ce50e973f73ed0bf85d12b0f8523d
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sun Aug 4 16:04:10 2013 -0400

    add < > <= >=

commit ff873f4441b026e8141bdc653b37498d67bdc08c
Merge: 2e5b302 0a11da5
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sun Aug 4 15:29:39 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

commit 2e5b302ac8ec0944afdc8a81cc6f887709331d8f
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sun Aug 4 15:29:11 2013 -0400
```

```
    Adding data library for note conversions

commit 0a11da5403bc054c7fba2638abd3d4f6b7d87656
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sun Aug 4 15:20:27 2013 -0400

    add PLUS, MINUS, EQUAL, NOTEQUAL

commit 944d4d64f64c7bfabc80b4a60b2f9dd7dfc0c896
Author: John <jcsizemo@gmail.com>
Date:   Sun Aug 4 14:46:49 2013 -0400

    Add two literals

commit e20b24cb15180b5a563ebdaba049d02a73e0d3a4
Author: John <jcsizemo@gmail.com>
Date:   Sun Aug 4 14:16:42 2013 -0400

    Add operator

commit 88199c708f859313f305bee56eb33ca2992c7f10
Author: Lauren Stephanian <lms2221@columbia.edu>
Date:   Sun Aug 4 13:10:15 2013 -0400

    Delete rhythmreadfile.ml

commit bd6edc36c1dbb0db0706880af1f51bfdcfb96664
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sun Aug 4 01:04:21 2013 -0400

    test file of ID plus

commit a27ce0e1a4c6989a4d1ffcfe375ce2217a21c847
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sun Aug 4 01:03:13 2013 -0400

    add plus operator for IDs

commit ef26275d910983f6d7be47369af6aca545cd5681
Author: Lauren Stephanian <lms2221@columbia.edu>
```

```
Date:   Sat Aug 3 16:35:27 2013 -0400

    Update rhythmreadfile.ml

commit 281ca1574b4c5154f959c530c1fc16d1231a28a5
Author: Lauren Stephanian <lms2221@columbia.edu>
Date:   Sat Aug 3 16:34:58 2013 -0400

    Rename rhythmread.ml to rhythmreadfile.ml

commit 4fc64bd5feb052b254fc5bc72b7b4dd8c346e4f8
Author: Lauren Stephanian <lms2221@columbia.edu>
Date:   Sat Aug 3 16:34:39 2013 -0400

    Create rhythmread.ml

commit 0cab387f0d77c12a2b1383e71ef02c199bcb6d12
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Fri Aug 2 22:56:57 2013 -0400

    Adding dump operator function

commit 4f736d0c445c0ce1f53f7b27382f855b3571bb9c
Author: John <John@JohnDesktop2.(none)>
Date:   Fri Aug 2 15:31:08 2013 -0400

    Cleaned up assignment function

commit 59259da04e3ed6434e4f0b4779ca28241795a384
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 2 02:10:43 2013 -0400

    Fixed print for variables within arrays

commit fb27dbbbbaa467db719e05132991baa76b69acd2
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 2 01:05:41 2013 -0400

    Cleaned up error messages

commit ee36a777c54472a34f8ad989347d5ab12ca318c6
```

```
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 2 01:01:06 2013 -0400

    Array assignment works

commit aba8bd79885111555aff524de49b32f1faa66731
Author: John <jcsizemo@gmail.com>
Date:   Fri Aug 2 00:09:17 2013 -0400

    Removed unnecessary files

commit 7dedea8377f46aa9112c754af50c01fca589aff3
Author: John <John@JohnDesktop2.(none)>
Date:   Thu Aug 1 18:44:32 2013 -0400

    Added array segment assignment - partially working

commit f411cba3b3dd9ae022f38a2f96a1a6ecfcd42d91
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Wed Jul 31 22:50:36 2013 -0400

    update ast.ml

commit e107416a35ebaac0756556fb255659bcd030b771
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Wed Jul 31 22:23:28 2013 -0400

    Adding function and variable support

commit 8b1304b0193376062abe74ea7e7ef5a2a3201ab8
Author: John <John@JohnDesktop2.(none)>
Date:   Wed Jul 31 15:11:15 2013 -0400

    Implemented array indexing

commit 2746e04d1552aad86757e4a1bb31a7d9bf2c9ac3
Author: John <John@JohnDesktop2.(none)>
Date:   Wed Jul 31 13:43:56 2013 -0400

    Cleaned up unnecessary files
```

```
commit 564550b623235709b202120b6f8aae4de9088a49
Author: John <John@JohnDesktop2.(none)>
Date:   Wed Jul 31 13:31:57 2013 -0400

    Added print statement, got assignment working

commit 080d3eb92507c6a19e1de1eac60b61a73b2541e2
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Tue Jul 30 22:01:06 2013 -0400

    Adding functions and programs

commit ab9e48b3f58bb79613da912e8b7ead982d6ca16b
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 30 16:54:12 2013 -0400

    delete test for

commit f7d6ecb3ac0d212f421b1c8e683d338543afe5ad
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 30 16:53:16 2013 -0400

    delete test for

commit dbcb3c0f28afc5f170c77b706e56253639cbd9fb
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 30 16:50:48 2013 -0400

    loop test file

commit 6a1ac995aa52ba30053a897dbb5c8d02eeed2906
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 30 16:45:57 2013 -0400

    loop test file

commit ff234ba6f8df7a4c0a8054bc0e4005eb5a8aeafa
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 30 16:42:10 2013 -0400

    batch file to test interpret
```

```
commit 49c88084d52de84179161011aa628671e1419f05
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 30 16:41:00 2013 -0400

    simple version of rhythm.ml to test interpret

commit 5d90ca4d6de17cf5c7e8387030068d0d0129a06b
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 30 16:37:34 2013 -0400

    now the statments work well

commit 882301bc4cfce86c2b4c9c43e60c4285cc23e134
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 30 16:34:54 2013 -0400

    update ast.ml add expression of loop

commit c6f65eca5487b27d3193c249e65656eb5c008a19
Author: John <John@JohnDesktop2.(none)>
Date:   Mon Jul 29 16:03:21 2013 -0400

    Trying to get note assignments to work

commit d949f97d3088ca09c5a6757362891a9c10e5ab14
Author: John <John@JohnDesktop2.(none)>
Date:   Mon Jul 29 15:42:06 2013 -0400

    Got variable assignment working

commit 9e5669fbebfb0f909169a49e7033fdba45968c49
Author: John <John@JohnDesktop2.(none)>
Date:   Mon Jul 29 14:47:53 2013 -0400

    Added some operation types to all files

commit 62d8ffd1362f8d21df90a9994eec630a4285b975
Author: John <John@JohnDesktop2.(none)>
Date:   Mon Jul 29 14:23:13 2013 -0400
```

```
    Got project working with Makefile

commit 131965e0fb956df88b67ef16e85ca5e9aecd2e06
Author: John <John@JohnDesktop2.(none)>
Date:   Mon Jul 29 14:11:45 2013 -0400

    Modified makefile, changed filenames

commit 87b6289d8395c9f51bfa40307d77960e3bfd408e
Author: John <John@JohnDesktop2.(none)>
Date:   Mon Jul 29 13:55:46 2013 -0400

    Changed microc.ml to rhythm.ml

commit 5b29c0cc166f0733a6100f9273d287ba8f6eec0c
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sun Jul 28 00:09:18 2013 -0400

    add temporal Loop function

commit b242f4ab20f446748d6c2de2968d81777a6337c8
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 23:58:13 2013 -0400

    temporary comment a part of code from Kai Comment to Kai to avoid an
error in compiling

commit 4fd3a88afc226e398dd6cf0a144ee6d2d7b7d773
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 23:34:52 2013 -0400

    delete For loop statement

commit b9c5de88a22eb7782e3e2602bdc7bf1ae8c32d84
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 12:49:45 2013 -0400

    Adding arrays

commit b4f5d806908ff4c98a3565c0116d52ad49a9c41c
Author: Yuankai Huo <huoyuankai@gmail.com>
```

```
Date:   Sat Jul 27 12:35:58 2013 -0400

    update

commit 66a73cea8ab2b598c1d2a62caeaa23ac8c0595b0
Author: John <jcsizemo@gmail.com>
Date:   Sat Jul 27 12:30:21 2013 -0400

    Got rid of unnecessary tokens

commit 4d8e55e15a16eb8a6b3ed37063af45a2568467dd
Author: John <jcsizemo@gmail.com>
Date:   Sat Jul 27 12:28:45 2013 -0400

    Added colon to scanner

commit a3e804d7899d6087a47fa2f028242b0fe6eaa6bc
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 12:23:42 2013 -0400

    add change in (parser.mly)

commit d0648e32d8f4981564e24e59b9cb35bf3fe9cfd6
Author: John <jcsizemo@gmail.com>
Date:   Sat Jul 27 12:16:18 2013 -0400

    Renamed Rhythm folder to src

commit 4d369f1daf6425806e160ee95884ecb90836cd78
Merge: 46e16df 93635f1
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 12:14:06 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

commit 46e16df697864d944cbbcd62bf554dc9b9692565
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 12:13:07 2013 -0400

    Updating parser to remove unused rules
```

```
commit 93635f1461e751be27d7745e5a03a4eb266ec77c
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 12:08:42 2013 -0400

    add change in (parser.mly)

commit de82743a1b2e24df00b1fb100dd7e01517ddfe71
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 11:58:00 2013 -0400

    add stmt Loop in (parser.mly and ast.mli)

commit b3f37407bbea07046ba44c90668a69fa20bab4f9
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 11:55:11 2013 -0400

    add stmt Loop in (parser.mly and ast.mli)

commit b2d8636ea2a76714e7a708d88e411d6d2afc30ad
Merge: 3e7ff46 3f29611
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 11:54:47 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

commit 3e7ff4642afad55693ad7142ab10c5bdc869056c
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 11:53:13 2013 -0400

    Updating ast and parser for notes

commit 3f296119777f7dcd45fd74bdc85b6f68a3e1980b
Merge: 0a345fb 743db3e
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 11:41:51 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

commit 743db3e4091ea83ea816dfb2793b67c8177bae41
Author: John <jcsizemo@gmail.com>
Date:   Sat Jul 27 11:40:13 2013 -0400
```

```
    Got rid of original files file

commit 0a345fb2eabb9cac44868a867d83bb891721a585
Merge: 21ee0b2 23701fd
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 11:29:01 2013 -0400

    Merge branch 'master' of https://github.com/jcsizemo/Rhythm

commit 21ee0b2463d8cf3684c4b58a5f6b599265f36f80
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Sat Jul 27 11:25:40 2013 -0400

    delete FOR loop in the statment in (parser.mly)

commit 23701fdd7bd0fffe444285556498e09dffca5915
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 11:23:14 2013 -0400

    Removing function variables

commit af34eb6280abe63981bb67777aae4a471ab5903f
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 10:59:08 2013 -0400

    Renaming ast.ml

commit 6c365c708279919300ab69efdf0a041d034126e9
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 10:56:51 2013 -0400

    Adding new file

commit c50418be25c8faaf2d3a617ca2b0758c7575d2b8
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sat Jul 27 10:46:17 2013 -0400

    Checking in fix

commit 633f9782104df56c1012aabf0b3840be24c6f3b2
```

```
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Thu Jul 25 23:15:46 2013 -0400

    add CSV.java 07.25.2013 11:15PM

commit dd773b106ad21d83e07977e79af02aaa04f755cb
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Thu Jul 25 23:14:20 2013 -0400

    add MessageInfo.java 07.25.2013 11:14PM

commit 2f44302eecd7ae192c947fcf4fcf532eb10eeb0b
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Thu Jul 25 23:13:47 2013 -0400

    add ListInstruments.java 07.25.2013 11:13PM

commit bb85394a2496ded149e1496edd5be1ff2ae170a9
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Thu Jul 25 23:08:50 2013 -0400

    add CSV.java 07.25.2013 11:08PM

commit 726eef139573abf11f5fc1e23c2990566fcbf7d0
Author: John <John@JohnDesktop2.(none)>
Date:   Thu Jul 25 17:46:41 2013 -0400

    Added tokens to parser, fixed errors in scanner

commit 926d74d5b28916f7944ab5f5968598b9a3617945
Author: John <John@JohnDesktop2.(none)>
Date:   Thu Jul 25 17:39:28 2013 -0400

    Added tokens to scanner

commit b7f1844a48c21e073f440c8374bcd0f64f7e3abd
Author: John <John@JohnDesktop2.(none)>
Date:   Thu Jul 25 17:37:32 2013 -0400

    Added tokens to scanner
```

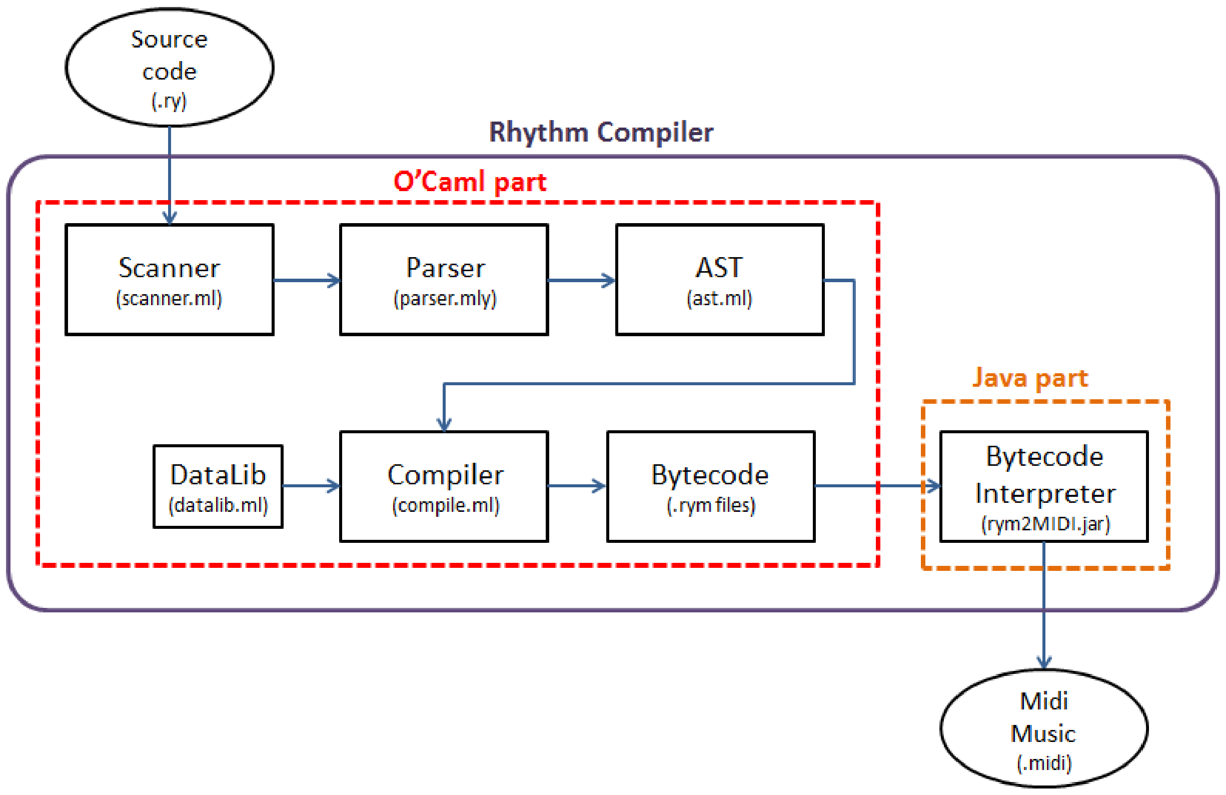
```
commit 851f31ec6f16585019d0b6f0fb71b91dfbe896ff
Author: John <jcsizemo@gmail.com>
Date:   Thu Jul 25 03:22:58 2013 -0400

    Put the MicroC files in the Rhythm directory to be modified into Rhythm
and replaced the compressed file with the original files with a directory
containing the original files

commit 5a16b59a1a2ec9c69969282bb3b5fc34f74c17d4
Author: John <jcsizemo@gmail.com>
Date:   Thu Jul 25 03:21:10 2013 -0400

    Realized I had cluttered the repository, got rid of scattered files

commit e020dcdb95c70b841d5f03398b694f904f8ab4a3
Author: John <jcsizemo@gmail.com>
Date:   Thu Jul 25 03:18:54 2013 -0400

    Added MicroC files to be modified into Rhythm

commit 2c3b3153147cc0bf6dc158a9d36201747e290ce7
Author: John <jcsizemo@gmail.com>
Date:   Thu Jul 25 03:18:16 2013 -0400

    Replaced old Ocaml files with a compressed version of the same files

commit 8be879efd7ade5cfc7fe1f4e850b18349acf82c2
Author: John <jcsizemo@gmail.com>
Date:   Thu Jul 25 03:17:08 2013 -0400

    Compressing original files

commit 94617eba82b0edc808062b013e96280de0766108
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 23 16:44:41 2013 -0400

    Create CSV2MIDI.java

    main function for generating MIDI from csv format.

commit 8d07c60b36225e6a390f1eab644ef31bb77898df
```

```
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Tue Jul 23 09:25:05 2013 -0400

    Revert "Create CSV2MIDI.java"

    This reverts commit 770ec4e308c09ee08b225c95094ce913d058dad2.

commit 770ec4e308c09ee08b225c95094ce913d058dad2
Author: Yuankai Huo <huoyuankai@gmail.com>
Date:   Mon Jul 22 16:48:03 2013 -0400

    Create CSV2MIDI.java

    main function to generate midi from java

commit 044ca64937732aa79cbb752d91916e85ea90f81c
Author: stauffec <cstauffer@CSTAUFFER-T410.corp.natpal.com>
Date:   Sun Jul 21 16:09:47 2013 -0400

    Adding base files

commit 2682b234d557dab9af21dca8fbdc70637464fd7f
Author: John <jcsizemo@gmail.com>
Date:   Wed Jul 10 23:13:57 2013 -0400

    Initial commit
```

5. Architectural Design

5.1. Architectural Design Diagram

The compiler takes Rhythm source code in suffix ".ry". It generates tokens and syntax tree by checking the lexical validations, syntax errors and semantic validations. After obtaining the tokens and the tree, Rhythm does lots of complicated operations per the operators, expressions, and statements. Then it generates the bytecode with suffix ".rym". Finally, a java based program is employed to interpret the bytecode into ".midi" music. The following figure illustrates the main components and steps of Rhythm projects:



5.2. Interface

5.2.1 Scanner

Rhythm utilizes a scanner and parser to read a program and generate the abstract syntax tree(AST). The scanner is implemented with Ocamllex which generates a list of valid tokens by reading .ry source code.

5.2.2 Parser

The parser, implemented by Ocamlyacc, accepts a sequence of tokens generated by the scanner. It only accepts legal syntaxes and rejects any illegal syntaxes. It then invokes the program routine to generate a list of variable/function declarations.

5.2.3 Abstract Syntax Tree (AST)

The AST first enumerates the tokens and interprets precedence and associativity between statements, operators and expressions. The AST primarily defines the core structure of Rhythm statements and expressions while the parser will then use to generate the AST.

5.2.4 Compiler

The compiler accepts the AST and recursively traverses each of its nodes. The result of doing so is a bytecode (.rym) file.

5.2.5 Data Library

Provides supplementary library functions which could be called by the compiler to perform certain tasks such as translating notes to integers, parsing duration information, etc.

5.2.6 Bytecode Interpreter

This is a java program that allows user to generate midi music files based on the bytecode. It traces through each track, chord and note, then generates midi files through bytecode reconstruction.

5.3 Responsibility

Scanner: John, Cris, Lauren, Kai
Parser: John, Cris, Lauren, Kai
AST: John, Cris, Lauren, Kai
DataLib: John, Cris
Compiler: John, Cris, Kai
Bytecode Compiler: Kai

6. Test Plan

6.1 Goals

The tests shown below were mainly chosen with the goal of covering every possible line of code in the code base and exploring every possibility when it came to Rhythm's expressions, operators, and statements. The purpose in doing so was to catch the subtle errors within the language and to determine if there were any inconsistencies for various possible inputs. The range of possibilities as permissible by the scanner were first determined in order to produce the test cases; after general test cases were written displaying basic functionality came boundary case analysis to discover program behavior in fringe cases. This methodology for producing tests was quite valuable in uncovering subtle bugs or odd behavior in execution. More detailed descriptions of the tests and what exactly they are testing are provided in the examples below. All tests are run by running "testall.sh" from the console. Each test will compare its output with an output file set aside for that particular test and will inform the user if any of them fail or other don't match up exactly.

Each test is located within the "test" folder within the "src" folder in Rhythm. The test itself as a "test-*.ry" file whereas the desired output is a ".out" file with the same prefix as the test.

Roles:

John:	Wrote the test cases for most of the operators and the arrays
Kai:	Wrote test cases checking code translation and file output
Lauren:	Wrote test cases for other operators and statements
Cris:	Wrote the tests for track recognition and the loops

6.2 Test Suite

6.2.1 Add Test

```
/* Addition Test

      The following tests test
      the following:
      1. Simple addition
      2. Note pitch change
      3. Chord pitch change
      4. Sequence pitch change with a chord
      5. Note mixing (this makes a chord)
      6. Note mixing (with left associativity,
            this first makes a chord with A and B
            and then makes two chords with A and C
            and B and C)
      7. Chord mixing (the LHS becomes three chords
            that each include D)
      8. Chord mixing (Elements sharing an index
                              will combine to make new chords)
*/

init() {
      print(1+1);
      print(A4 + 1);
      print([A4,B4] + 1);
      print([A4,B4,[C4,D4]] + 1);
      print(A4 + B4);
      print(A4 + B4 + C4);
      print([A4,B4,C4] + D4);
      print([A4,B4,C4] + [D4,E4]);
}
```

Output:

```
2
A#4.4
[A#4.4,C4.4]
[A#4.4,C4.4,[C#4.4,D#4.4]]
[A4,B4]
[[A4,C4],[B4,C4]]
[[A4,D4],[B4,D4],[C4,D4]]
[[A4,D4],[B4,E4],C4]
```

6.2.2 Arpeggio Test

```
/*    Arpeggio Test

      Tests the following:
      1. Algorithms in Rhythm
      2. Loop loop to build track
      3. If the songs are building with the correct notes
*/

init() {
      c = C4;
      song = [];
      loop (0 : 4) {
            song ::= c;
            c += 4;
      }
      song[2]--;
      song ::= song[2] :: song[1] :: song[0];
      print(song);
}
```

Output:

```
[C4,E4.4,G4.4,C5.4,G4.4,E4.4,C4]
```

Output:

```
[C4,E4.4,G4.4,C5.4,G4.4,E4.4,C4]
```

6.2.3 Array Test

```
/*    Array Test

      The following three tests
      test the following:
      1. Simple single element array
      2. Multi-element array
      3. Nested array
*/

init() {
      print([1]);
      print([1,2,3]);
      print([1,2,[1,2,3]]);
}
```

Output:

```
[1]
[1,2,3]
[1,2,[1,2,3]]
```

6.2.4 Concatenation Test

```
/*    Concatentation Test

      Tests the following:
      1. Two notes into a sequence
      2. Note to end of a sequence
      3. Chord to end of sequence
*/

init() {

      print(A4 :: A4);
      print([A4,B4] :: C4);
      print([A4,B4] :: [[C4,D4]]);

}
```

Output:

```
[A4,A4]
[A4,B4,C4]
[A4,B4,[C4,D4]]
```

6.2.5 Equality Test

```
/*    Equality Test

      Tests the following:

      1. Note equal to itself (A4.4 and A4 should be same)
      2. Literal equal to itself
      3. Rest equal to itself
      4. Different notes
      5. Different durations
      6. Different literals
      7. Literal to note
      8. Chord to itself
      9. Chord to literal
      10. Same sequences
      11. Different sequences
*/
```

```
init() {
	print(A4.4 == A4);
	print(1 == 1);
	print(R == R);
	print(A4 == B4);
	print(A4.4 == A4.8);
	print(1 == 2);
	print(1 == A4);
	print([A4,B4] == [A4,B4]);
	print([A4,B4] == 1);
	print([A4,B4,[C4,D4,E4]] == [A4,B4,[C4,D4,E4]]);
	print([A4,B4,[C4,D4,E4]] == [A4,B4,[C4,D4,F4]]);
}
```

Output:

```
1
1
1
0
0
0
0
1
0
1
0
```

6.2.6 Expression Test

```
/*	Expression Test

	Tests the following:
	1. Operator associativity and precedence
	2. All operator functionality with a statement
	3. Assignments
*/

init() {
	print(1 + 2 - 3);
	a = [A4,B4,C4];
	b = A4 - (1 + 2) * 4 / 2 :: [[a[1]+1,B4]] -> 3 >> 3 << 1;
```

```
      print(b);
      b++;
      print(b);
      b--;
      print(b);
      b >>= 2;
      print(b);
      b <<= 3;
      print(b);
      b += 1;
      print(b);
      b -= 2;
      print(b);
      b ::= G#7;
      print(b);
      b *= 2;
      print(b);
      b /= 4;
      print(b);
}
```

Output:

```
0
[F#3.2,[C6.4,B6.4],[C6.4,B6.4],[C6.4,B6.4]]
[G3.2,[C#6.4,C6.4],[C#6.4,C6.4],[C#6.4,C6.4]]
[F#3.2,[C6.4,B6.4],[C6.4,B6.4],[C6.4,B6.4]]
[F#5.2,[C8.4,B8.4],[C8.4,B8.4],[C8.4,B8.4]]
[F#2.2,[C5.4,B5.4],[C5.4,B5.4],[C5.4,B5.4]]
[G2.2,[C#5.4,C5.4],[C#5.4,C5.4],[C#5.4,C5.4]]
[F2.2,[B5.4,A#5.4],[B5.4,A#5.4],[B5.4,A#5.4]]
[F2.2,[B5.4,A#5.4],[B5.4,A#5.4],[B5.4,A#5.4],G#7]
[F2.1,[B5.2,A#5.2],[B5.2,A#5.2],[B5.2,A#5.2],G#7.2]
[F2.4,[B5.8,A#5.8],[B5.8,A#5.8],[B5.8,A#5.8],G#7.8]
```

6.2.7 File Test

```
/*    File Test

      The following tests test the following:

      1. File prints OK
*/
```

```
def s;


track_foo()
{
  def c;
  c = [[A0.16,A1.16,A2.16],A3,A4.16,R.8,A2];
  return c;
}

track_foo2()
{
  return s;
}

init()
{
  s = [A5,B3,R.1,D7];
}
```

Output:

```
RYM file created successfully
RYM file created successfully
```

6.2.8 Function Test

```
/*    Function Test

      Tests the following:
      1. Functionality of functions
      2. Argument passing
      3. Result returning
*/

getNotes() {
      return [A4,B4,C4];
}

plusOne(e1) {
      return e1 + 1;
}
```

```
add(e1,e2) {
      return e1 + e2;
}


init() {
      a = getNotes();
      b = plusOne(a);
      c = add(a,b);
      print(a);
      print(b);
      print(c);
}
```

Output:

```
[A4,B4,C4]
[A#4.4,C4.4,C#4.4]
[[A4,A#4.4],[B4,C4.4],[C4,C#4.4]]
```

6.2.9 Greater Than Or Equal Test

```
/*    Greater Than Or Equal Test

      Tests the following:
      1. RHS one pitch lower
      2. Notes equal
      3. RHS one pitch higher
*/

init() {

      print(B4 >= A#4);
      print(B4 >= B4);
      print(B4 >= C4);
}
```

Output:

```
1
1
```

```
0
```

6.2.10 Global Variable Test

```
/*	Global Test

	Tests the following:
	1. Checks to see if the assignment to
	the global within the function works
	or goes out of scope
*/

def c;

init() {
	c = [A4,B4,C4];
	testGlobal();
}

testGlobal() {
	print(c);
}
```

Output:

```
[A4,B4,C4]
```

6.2.11 Greater Than Test

```
/*	Greater Than Test

	Tests the following:
	1. RHS one pitch lower
	2. Notes equal
	3. RHS one pitch higher
*/

init() {

	print(B4 > A#4);
	print(B4 > B4);
```

```
      print(B4 > C4);

}
```

Output:

```
1
0
0
```

6.2.12 If Test

```
/*    If Test

      Tests the following:
      1. True condition
      2. False condition
      3. Nested if statements
*/

init() {

      if (A4 == A4) {
            print(1);
      }
      else {
            print(2);
      }

      if (A4 != A4) {
            print(3);
      }
      else {
            print(4);
      }

      if (A4 == A4) {
            print(5);
      }

      if (A4 == A4) {
            if (A4 == A4) {
                  print(6);
            }
      }

      if (A4 == A4) {
            if (A4 != A4) {
                  print(7);
            }
            else {
                  print(8);
            }
      }
}
```

Output:

```
1
4
5
6
8
```

6.2.13 Index Test

```
/*    Index Test

      Tests the following:
      1. 1D index
      2. 2D index
      3. Variable index
*/

init() {

      a = [A4,B4,C4];
      print(a[1]);
      a = [A4,B4,[C4,D4,E4]];
      print(a[2][1]);
      c = 1;
      print(a[c]);

}
```

Output:

```
B4
D4
B4
```

6.2.14 Lengthen Test

```
/*    Lengthen Test

      Tests the following:
      1. Lengthen basic note
      2. Lengthen note with duration
      3. Sixteenth to whole
      4. Duration change across chord
      5. Duration change with complex notes
*/

init() {
      print(A4 * 2);
      print(A4.16 * 2);
      print(A4.16 * 16);
      print([A4,B4,C4] * 2);
      print([Ab4.8,B4.8,C#4.8] * 2);
}
```

Output:

```
A4.2
A4.8
A4.1
[A4.2,B4.2,C4.2]
[Ab4.4,B4.4,C#4.4]
```

6.2.15 Less Than Or Equal Test

```
/*    Less Than Or Equal Test

      Tests the following:
      1. RHS one pitch lower
      2. Notes equal
      3. RHS one pitch higher
```

```
*/

init() {

      print(B4 <= A#4);
      print(B4 <= B4);
      print(B4 <= C4);
}
```

Output:

```
0
1
1
```

6.2.16 Literal Test

```
/*    Literal Test

      Tests the following:
      1. Positive integers
      2. Negative integers
      3. Zero
*/

init() {

      print(1);
      print(0);
      print(-1);
      print(100);
      print(-100);
}
```

Output:

```
1
0
-1
100
-100
```

6.2.17 Local Variable Test

```
/*    Local Var Test

      Tests the following:
      1. Print local variable
      2. Print global variable:
            Check to see local version
            is not assigned to global version
*/

def c;

foo(c) {
      c--;
      print(c);
      bar();
}

bar() {
      print(c);
}


init() {
      c = C4.4;
      foo(c);
}
```

Output:

```
B4.4
C4.4
```

6.2.18 Loop Test

```
/*    Loop Test

      Tests the following:
      1. Tests loop to build a song
            with all halfsteps up
            to a new octave
```

```
*/

init() {

      allNotes = [];
      start = A4;

      loop(0:12) {
            allNotes ::= start;
            start++;
      }

      print(allNotes);

}
```

Output:

```
[A4,A#4.4,B4.4,C4.4,C#4.4,D4.4,D#4.4,E4.4,F4.4,F#4.4,G4.4,G#4.4]
```

6.2.19 Less Than Test

```
/*    Less Than Test

      Tests the following:
      1. RHS one pitch lower
      2. Notes equal
      3. RHS one pitch higher
*/

init() {

      print(B4 < A#4);
      print(B4 < B4);
      print(B4 < C4);
}
```

Output:

```
0
0
1
```

6.2.20 Minus Test

```
/*    Minus Test

      Tests the following:
      1. Unary negation
      2. Positive minus positive
      3. Positive minus negative
      5. Pitch decrease
      6. Chord pitch decrease
      7. Pitch decrease across sequence
*/

init() {
      print(-1 - 1);
      print(43 - 7);
      print(43 - -7);
      print(A4 - 1);
      print([C4,F4] - 1);
      print([A4.1,B4.2,[C4.4,D4.8]] - 1);
}
```

Output:

```
-2
36
50
G#3.4
[B4.4,E4.4]
[G#3.1,A#4.2,[B4.4,C#4.8]]
```

6.2.21 Note Test

```
/*    Note Test

      Tests the following:
      1. Basic note
      2. Sharp symbol
      3. Flat symbol
      4. Sharp/flat symbol and duration
*/
```

```
init() {

      print(G4);
      print(G#4);
      print(Gb4);
      print(Gb4.4);
}
```

Output:

```
G4
G#4
Gb4
Gb4.4
```

6.2.22 Octave Down Test

```
/*    Octave Down Test

      Tests the following:
      1. Note shifted down two octaves
*/

init() {

      print(C4 << 2);
}
```

Output:

```
C2.4
```

6.2.23 Octave Up Test

```
/*    Octave Up Test

      Tests the following:
      1. Shift two octaves up (C6)
*/
```

```
init() {

	print(C4 >> 2);
}
```

Output:

```
C6.4
```

6.2.24 Rest Test

```
/*	Rest Test

	Tests the following:
	1. Rest without defined duration
	2. Rest with defined duration
*/

init() {

	print(R);
	print(R.8);

}
```

Output:

```
R
R.8
```

6.2.25 Scale Test

```
/*	Scale Test

	Tests the following:
	1. Algorithms in Rhythm
	2. While loop to index array
	3. Track building
	4. If the songs are building with the correct notes
```

```
*/

/* major scales are whole, whole, half, whole, whole, whole, half */
/* minor scale: whole half whole whole half whole whole */

init() {
      song = [];
      c = C4;
      steps = [2,2,1,2,2,2,1,1];
      count = 0;
      while(count < 8) {
            song ::= c;
            c += steps[count];
            count++;
      }
      print(song);

      song = [];
      c = E4;
      steps = [2,1,2,2,1,2,2,2];
      count = 0;
      while(count < 8) {
            song ::= c;
            c += steps[count];
            count++;
      }
      print(song);
}
```

Output:

```
[C4,D4.4,E4.4,F4.4,G4.4,A5.4,B5.4,C5.4]
[E4,F#4.4,G4.4,A5.4,B5.4,C5.4,D5.4,E5.4]
```

6.2.26 Shorten Test

```
/*    Shorten Test

      Tests the following:
      1. Quarter to eighth notes
      2. Whole to sixteenth
      3. Chord shortening
      4. Sequence shortening
```

```
*/

init() {

      print(A4 / 2);
      print(A4.1 / 16);
      print([A4,B4,C4] / 2);
      print([Ab4.8,B4.8,[C#4.8,D8]] / 2);

}
```

Output:

```
A4.8
A4.16
[A4.8,B4.8,C4.8]
[Ab4.16,B4.16,[C#4.16,D8.8]]
```

6.2.27 Stretch Test

```
/*    Stretch Test

      Tests the following:
      1. Note to 4 notes
      2. Two note sequence repeated 4 times
      3. 3 repeated chords
*/

init() {

      print(A4 -> 3);
      print([A4,B4] -> 3);
      print([[A4,B4]] -> 3);
}
```

Output:

```
[A4,A4,A4]
[A4,B4,A4,B4,A4,B4]
[[A4,B4],[A4,B4],[A4,B4]]
```

6.2.28 Unequal Test

```
/*    Unequal Test

      Tests the following:

      1. Note equal to itself (A4.4 and A4 should be same)
      2. Literal equal to itself
      3. Rest equal to itself
      4. Different notes
      5. Different durations
      6. Different literals
      7. Literal to note
      8. Chord to itself
      9. Chord to literal
      10. Same sequences
      11. Different sequences
*/

init() {

      print(A4.4 != A4);
      print(1 != 1);
      print(R != R);
      print(A4 != B4);
      print(A4.4 != A4.8);
      print(1 != 2);
      print(1 != A4);
      print([A4,B4] != [A4,B4]);
      print([A4,B4] != 1);
      print([A4,B4,[C4,D4,E4]] != [A4,B4,[C4,D4,E4]]);
      print([A4,B4,[C4,D4,E4]] != [A4,B4,[C4,D4,F4]]);


}
```

Output:

```
0
0
0
1
1
1
```

```
1
0
1
0
1
```

6.2.29 While Test

```
init() {
      print(1+1);
      print(1+2+3);
      print(A4 + 1);
      print(A4 + 1 + 2);
      print([A4,B4] + 1);
      print([A4,B4] + 1 + 2);
      print([A4,B4,[C4,D4]] + 1);
      print(A4 + B4);
      print(A4 + B4 + C4);
      print([A4,B4,C4] + D4);
}
```

Output:

```
0
1
2
3
4
5
6
7
8
9
```

6.3 Sample Programs

6.3.1 Arpeggio

```
track_cArp() {
      c = C4;
      song = [];
      loop (0 : 4) {
            song ::= c;
            c += 4;
      }
      song[2]--;
      song ::= song[2] :: song[1] :: song[0];
      return song;
}
```

Output:

```
Track: cArp
0 51
1 51
2 51
3 51
4 55
5 55
6 55
7 55
8 58
9 58
10 58
11 58
12 63
13 63
14 63
15 63
16 58
17 58
18 58
19 58
20 55
21 55
22 55
23 55
24 51
25 51
26 51
```

```
27 51
```

6.3.2 Scales

```
/* major scales are whole, whole, half, whole, whole, whole, half */
/* minor scale: whole half whole whole half whole whole */

cMajor() {
      song = [];
      c = C4;
      steps = [2,2,1,2,2,2,1,1];
      count = 0;
      while(count < 8) {
            song ::= c;
            c += steps[count];
            count++;
      }
      return song;
}

track_cMajor_and_eMinor() {
      song = cMajor() :: R.1->2;
      c = E4;
      steps = [2,1,2,2,1,2,2,2];
      count = 0;
      while(count < 8) {
            song ::= c;
            c += steps[count];
            count++;
      }
      return song;
}
```

Output:

```
Track: cMajor_and_eMinor
0 51
1 51
2 51
3 51
4 53
5 53
6 53
7 53
```

```
8 55
9 55
10 55
11 55
12 56
13 56
14 56
15 56
16 58
17 58
18 58
19 58
20 60
21 60
22 60
23 60
24 62
25 62
26 62
27 62
28 63
29 63
30 63
31 63
64 55
65 55
66 55
67 55
68 57
69 57
70 57
71 57
72 58
73 58
74 58
75 58
76 60
77 60
78 60
79 60
80 62
81 62
82 62
83 62
84 63
85 63
86 63
```

```
87 63
88 65
89 65
90 65
91 65
92 67
93 67
94 67
95 67
```

6.3.3 Row Row Row Your Boat

```
getBaseNotes() {
    def row;
    def rowbase;
    rowbase = [[C5,E5,G5], [C5,E5,G5], [C5,E5,G5],D5.8, E5.8, E5.8, D5.8,
E5.8, F5.8, G5.2, C6, G5, E5, C5, G5.8, F5.8, E5.8, D5.8, [C5,E5,G5]];

   row = rowbase->3;
   return row;
}

track_1() {
    return getBaseNotes();
}

track_2() {
    return R.1->4 :: getBaseNotes() << 2;
}

track_3() {
    return R.1->2 :: getBaseNotes() << 1;
}
```

Output:

```
Track: 1
0 63
1 63
2 63
3 63
0 67
1 67
2 67
```

```
3 67
0 70
1 70
2 70
3 70
4 63
5 63
6 63
7 63
4 67
5 67
6 67
7 67
4 70
5 70
6 70
7 70
8 63
9 63
10 63
11 63
8 67
9 67
10 67
11 67
8 70
9 70
10 70
11 70
12 65
13 65
14 67
15 67
16 67
17 67
18 65
19 65
20 67
21 67
22 68
23 68
24 70
25 70
26 70
27 70
28 70
29 70
```

```
30 70
31 70
32 75
33 75
34 75
35 75
36 70
37 70
38 70
39 70
40 67
41 67
42 67
43 67
44 63
45 63
46 63
47 63
48 70
49 70
50 68
51 68
52 67
53 67
54 65
55 65
56 63
57 63
58 63
59 63
56 67
57 67
58 67
59 67
56 70
57 70
58 70
59 70
60 63
61 63
62 63
63 63
60 67
61 67
62 67
63 67
60 70
```

```
61 70
62 70
63 70
64 63
65 63
66 63
67 63
64 67
65 67
66 67
67 67
64 70
65 70
66 70
67 70
68 63
69 63
70 63
71 63
68 67
69 67
70 67
71 67
68 70
69 70
70 70
71 70
72 65
73 65
74 67
75 67
76 67
77 67
78 65
79 65
80 67
81 67
82 68
83 68
84 70
85 70
86 70
87 70
88 70
89 70
90 70
91 70
```

```
92 75
93 75
94 75
95 75
96 70
97 70
98 70
99 70
100 67
101 67
102 67
103 67
104 63
105 63
106 63
107 63
108 70
109 70
110 68
111 68
112 67
113 67
114 65
115 65
116 63
117 63
118 63
119 63
116 67
117 67
118 67
119 67
116 70
117 70
118 70
119 70
120 63
121 63
122 63
123 63
120 67
121 67
122 67
123 67
120 70
121 70
122 70
```

```
123 70
124 63
125 63
126 63
127 63
124 67
125 67
126 67
127 67
124 70
125 70
126 70
127 70
128 63
129 63
130 63
131 63
128 67
129 67
130 67
131 67
128 70
129 70
130 70
131 70
132 65
133 65
134 67
135 67
136 67
137 67
138 65
139 65
140 67
141 67
142 68
143 68
144 70
145 70
146 70
147 70
148 70
149 70
150 70
151 70
152 75
153 75
```

```
154 75
155 75
156 70
157 70
158 70
159 70
160 67
161 67
162 67
163 67
164 63
165 63
166 63
167 63
168 70
169 70
170 68
171 68
172 67
173 67
174 65
175 65
176 63
177 63
178 63
179 63
176 67
177 67
178 67
179 67
176 70
177 70
178 70
179 70

Track: 2
64 39
65 39
66 39
67 39
64 43
65 43
66 43
67 43
64 46
65 46
66 46
```

```
67 46
68 39
69 39
70 39
71 39
68 43
69 43
70 43
71 43
68 46
69 46
70 46
71 46
72 39
73 39
74 39
75 39
72 43
73 43
74 43
75 43
72 46
73 46
74 46
75 46
76 41
77 41
78 43
79 43
80 43
81 43
82 41
83 41
84 43
85 43
86 44
87 44
88 46
89 46
90 46
91 46
92 46
93 46
94 46
95 46
96 51
97 51
```

```
98 51
99 51
100 46
101 46
102 46
103 46
104 43
105 43
106 43
107 43
108 39
109 39
110 39
111 39
112 46
113 46
114 44
115 44
116 43
117 43
118 41
119 41
120 39
121 39
122 39
123 39
120 43
121 43
122 43
123 43
120 46
121 46
122 46
123 46
124 39
125 39
126 39
127 39
124 43
125 43
126 43
127 43
124 46
125 46
126 46
127 46
128 39
```

```
129 39
130 39
131 39
128 43
129 43
130 43
131 43
128 46
129 46
130 46
131 46
132 39
133 39
134 39
135 39
132 43
133 43
134 43
135 43
132 46
133 46
134 46
135 46
136 41
137 41
138 43
139 43
140 43
141 43
142 41
143 41
144 43
145 43
146 44
147 44
148 46
149 46
150 46
151 46
152 46
153 46
154 46
155 46
156 51
157 51
158 51
159 51
```

```
160 46
161 46
162 46
163 46
164 43
165 43
166 43
167 43
168 39
169 39
170 39
171 39
172 46
173 46
174 44
175 44
176 43
177 43
178 41
179 41
180 39
181 39
182 39
183 39
180 43
181 43
182 43
183 43
180 46
181 46
182 46
183 46
184 39
185 39
186 39
187 39
184 43
185 43
186 43
187 43
184 46
185 46
186 46
187 46
188 39
189 39
190 39
```

```
191 39
188 43
189 43
190 43
191 43
188 46
189 46
190 46
191 46
192 39
193 39
194 39
195 39
192 43
193 43
194 43
195 43
192 46
193 46
194 46
195 46
196 41
197 41
198 43
199 43
200 43
201 43
202 41
203 41
204 43
205 43
206 44
207 44
208 46
209 46
210 46
211 46
212 46
213 46
214 46
215 46
216 51
217 51
218 51
219 51
220 46
221 46
```

```
222 46
223 46
224 43
225 43
226 43
227 43
228 39
229 39
230 39
231 39
232 46
233 46
234 44
235 44
236 43
237 43
238 41
239 41
240 39
241 39
242 39
243 39
240 43
241 43
242 43
243 43
240 46
241 46
242 46
243 46

Track: 3
32 51
33 51
34 51
35 51
32 55
33 55
34 55
35 55
32 58
33 58
34 58
35 58
36 51
37 51
38 51
```

```
39 51
36 55
37 55
38 55
39 55
36 58
37 58
38 58
39 58
40 51
41 51
42 51
43 51
40 55
41 55
42 55
43 55
40 58
41 58
42 58
43 58
44 53
45 53
46 55
47 55
48 55
49 55
50 53
51 53
52 55
53 55
54 56
55 56
56 58
57 58
58 58
59 58
60 58
61 58
62 58
63 58
64 63
65 63
66 63
67 63
68 58
69 58
```

```
70 58
71 58
72 55
73 55
74 55
75 55
76 51
77 51
78 51
79 51
80 58
81 58
82 56
83 56
84 55
85 55
86 53
87 53
88 51
89 51
90 51
91 51
88 55
89 55
90 55
91 55
88 58
89 58
90 58
91 58
92 51
93 51
94 51
95 51
92 55
93 55
94 55
95 55
92 58
93 58
94 58
95 58
96 51
97 51
98 51
99 51
96 55
```

```
97 55
98 55
99 55
96 58
97 58
98 58
99 58
100 51
101 51
102 51
103 51
100 55
101 55
102 55
103 55
100 58
101 58
102 58
103 58
104 53
105 53
106 55
107 55
108 55
109 55
110 53
111 53
112 55
113 55
114 56
115 56
116 58
117 58
118 58
119 58
120 58
121 58
122 58
123 58
124 63
125 63
126 63
127 63
128 58
129 58
130 58
131 58
```

```
132 55
133 55
134 55
135 55
136 51
137 51
138 51
139 51
140 58
141 58
142 56
143 56
144 55
145 55
146 53
147 53
148 51
149 51
150 51
151 51
148 55
149 55
150 55
151 55
148 58
149 58
150 58
151 58
152 51
153 51
154 51
155 51
152 55
153 55
154 55
155 55
152 58
153 58
154 58
155 58
156 51
157 51
158 51
159 51
156 55
157 55
158 55
```

```
159 55
156 58
157 58
158 58
159 58
160 51
161 51
162 51
163 51
160 55
161 55
162 55
163 55
160 58
161 58
162 58
163 58
164 53
165 53
166 55
167 55
168 55
169 55
170 53
171 53
172 55
173 55
174 56
175 56
176 58
177 58
178 58
179 58
180 58
181 58
182 58
183 58
184 63
185 63
186 63
187 63
188 58
189 58
190 58
191 58
192 55
193 55
```

```
194 55
195 55
196 51
197 51
198 51
199 51
200 58
201 58
202 56
203 56
204 55
205 55
206 53
207 53
208 51
209 51
210 51
211 51
208 55
209 55
210 55
211 55
208 58
209 58
210 58
211 58
```

7. Lessons Learned

John Sizemore

Arrays are very, very annoying to program into a language. VERY annoying. Especially if you are trying to index them. I realize what the teacher meant at the beginning of the semester when he was talking about students “seeing the light” about OCaml towards the end; at the beginning I absolutely hated everything about it but now I can see its charm, its power, and furthermore how it could make the job of writing a compiler easier.

I think that programming languages evolve with use; there was no way we would get it right on the first try. It was only when we played around with what we had made that we realized what we should be adding instead of what we thought we should be adding.

I have a learned a lot about programming languages and I feel like I know more about what it means for a language to “be” a language. I feel like in the future I will write code or at least look at code differently because of this class.

I would not change anything about the way the summer session has progressed with my team. The team I had the chance to work with was great in every aspect. We were able to finish our project quickly and easily with minimal stress. All team members were on board from the beginning, and our work on the project was very consistent over the past 6 weeks. We definitely did not try to cram everything together at the end. For that I am very grateful and to future teams I can say that you had better work consistently on the project throughout the semester and you had better split your roles into modular tasks that can be completed concurrently. Furthermore, START EARLY. I can easily see how this project could turn into a nightmare with a late start.

Cristopher Stauffer

I learned that often a mix of vertical tasks (e.g. notes from AST->Midi) as well as horizontal tasks (e.g. research note arithmetic, research midi format) worked well in this case.

I also learned that for a programming language the best place to start is a blank sheet of paper. This forces you to justify every element of your language, without just “expecting” things to work.

Lauren Stephanian

An early start is extremely beneficial
Weekly meetings and maintaining communication are very important
Modular division of tasks critical

Yuankai Huo

After completing this project, I learned so much in three main aspects.
First, technical level. I have deeper comprehension on program language and compiler. Program language to engineers is like car to people. I'm excited that now I'm not only know how to drive a car, but also build a car.

Second, project experience. It's really helpful for me to participate a whole large-scale software development, from proposal, LRM, implementation to final report. More important, we need to use unfamiliar software O'caml rather than C and Java. This cultivates my ability to handle a new language as soon as possible.

Last and most important, team work. In industry, it's almost impossible to accomplish a complicate task by ourselves. During the project, we learned from each other, improved trust to each other and took full advantage of every member's strong points.

Thanks very much to John, Cris and Lauren. It's impossible to finish the project without you guys. Working with you guys is the most beautiful memory in my summer 2013. Thanks very much to Professor Edwards. We really appreciate that you spend lots of time on our project and give us a lot of helpful inputs.

8. Appendix

8.1 source codes

8.1.1 scanner.mll

```
{ open Parser }

rule token = parse
  [' ' '\t' '\r' '\n'] { token lexbuf } (* Whitespace *)
| "/*"                 { comment lexbuf }      (* Comments *)
| '('                  { LPAREN }
| ')'                  { RPAREN }
| '['                  { LBRACKET }
| ']'                  { RBRACKET }
| '{'                  { LBRACE }
| '}'                  { RBRACE }
| ';'                  { SEMI }
| ','                  { COMMA }
| '+'                  { PLUS }
| '-'                  { MINUS }
| '='                  { ASSIGN }
| ':'                  { COLON }
| "::"                 { CONCAT }
| '<'                  { LT }
| "<="                 { LEQ }
| ">"                  { GT }
| ">="                 { GEQ }
| "=="                 { EQ }
| "!="                 { NEQ }
| "<<"                 { OCTDOWN }
| ">>"                 { OCTUP }
| "++"                 { HALFUP }
| "--"                 { HALFDOWN }
| "+="                 { ASSIGN_PLUS }
| "-="                 { ASSIGN_MINUS }
| ">>="                { ASSIGN_OCTUP }
| "<<="                { ASSIGN_OCTDOWN }
| "::="                { ASSIGN_CONCAT }
| "*="                 { ASSIGN_INC }
| "/="                 { ASSIGN_DEC }
| "*"                  { INCREASE_DURATION }
| "/"                  { DECREASE_DURATION }
| "->"                 { STRETCH }
| "if"                 { IF }
| "else"               { ELSE }
| "while"              { WHILE }
| "return"             { RETURN }
| "loop"               { LOOP }
| "true"               { TRUE }
| "false"              { FALSE }
| "def"                { DEF }
| ['-']?['0'-'9']+ as lxm { LITERAL(int_of_string lxm) }
| ['A' - 'G']['b' '#']?['0'-'9'] as lxm { NOTE(lxm) }
| ['A' - 'G']['b' '#']?['0'-'9']('.'['0'-'9']+)? as lxm { NOTE(lxm) }
| 'R'('.'['0'-'9']+)? as lxm { REST(lxm) }
| ['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']* as lxm { ID(lxm) }
| eof { EOF }
| _ as char { raise (Failure("illegal character " ^ Char.escaped char )) }

and comment = parse
  "*/" { token lexbuf }
| _    { comment lexbuf }
```

8.1.2 parser.mly

```
%{ open Ast %}

%token SEMI LPAREN RPAREN LBRACE RBRACE COMMA LBRACKET RBRACKET
%token COLON
%token PLUS MINUS ASSIGN CONCAT STRETCH
%token EQ NEQ OCTDOWN OCTUP HALFUP HALFDOWN
%token LT LEQ GT GEQ

%token INCREASE_DURATION DECREASE_DURATION
%token ASSIGN_PLUS ASSIGN_MINUS ASSIGN_REMOVE
%token ASSIGN_OCTUP ASSIGN_OCTDOWN
%token ASSIGN_INC ASSIGN_DEC
%token ASSIGN_CONCAT
%token RETURN IF ELSE WHILE LOOP
%token CLOSEFILE OPENFILE TRUE FALSE TEMPO
%token STARTTRACK STOPTRACK
%token <int> LITERAL
%token <string> ID
%token <string> NOTE
%token <string> REST
%token EOF
%token  DEF

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN ASSIGN_PLUS ASSIGN_MINUS ASSIGN_OCTUP ASSIGN_OCTDOWN ASSIGN_CONCAT
ASSIGN_INC ASSIGN_DEC
%left EQ NEQ
%left LT GT LEQ GEQ
%left CONCAT
%left PLUS MINUS OCTUP OCTDOWN INCREASE_DURATION DECREASE_DURATION STRETCH
%right HALFUP HALFDOWN

%start program
%type <Ast.program> program

%%
/* program: just a set of functions */
program:
   /* nothing */ { [], [] }
 | program vdecl { ($2 :: fst $1), snd $1 }
 | program fdecl { fst $1, ($2 :: snd $1) }

fdecl:
   ID LPAREN formals_opt RPAREN LBRACE vdecl_list stmt_list RBRACE
       { { fname = $1;
   formals = $3;
   locals = List.rev $6;
   body = List.rev $7 } }

/* declared function args */
formals_opt:
       /* nothing */ { [] }
  | formal_list   { List.rev $1 }

formal_list:
       ID                    { [$1] }
  | formal_list COMMA ID { $3 :: $1 }

vdecl_list:
       /* nothing */ { [] }
  | vdecl_list vdecl { $2 :: $1 }

vdecl:
   DEF ID SEMI { $2 }

stmt_list:
       /* nothing */  { [] }
  | stmt_list stmt { $2 :: $1 }

stmt:
       expr SEMI { Expr($1) }
  | RETURN expr SEMI { Return($2) }
  | LBRACE stmt_list RBRACE { Block(List.rev $2) }
  | IF LPAREN expr RPAREN stmt %prec NOELSE { If($3, $5, Block([])) }
  | IF LPAREN expr RPAREN stmt ELSE stmt        { If($3, $5, $7) }
  | WHILE LPAREN expr RPAREN stmt { While($3, $5) }
  | LOOP LPAREN expr COLON expr RPAREN stmt { Loop($3, $5, $7) }

expr:
       LITERAL       { Literal($1) }
  | ID               { Id($1) }
  | NOTE             { Note($1) }
  | REST             { Rest($1) }
  | expr PLUS   expr { Binop($1, Plus,   $3) }
  | expr MINUS  expr { Binop($1, Minus,   $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 CONCAT expr { Binop($1, Concat, $3) }
  | expr ASSIGN expr   { Assign($1, $3) }
  | expr HALFUP { Assign($1, Binop($1, Plus, Literal(1)))}
  | expr HALFDOWN { Assign($1, Binop($1, Minus, Literal(1)))}
  | expr OCTUP expr   {Binop($1,Octup,$3)}
  | expr OCTDOWN expr   { Binop($1, Octdown, $3)}
  | expr INCREASE_DURATION expr { Binop($1, IncDuration, $3)}
  | expr DECREASE_DURATION expr { Binop($1, DecDuration, $3)}
  | expr ASSIGN_PLUS expr { Assign($1, Binop($1,Plus,$3) )}
  | expr ASSIGN_MINUS expr { Assign($1, Binop($1,Minus,$3) )}
  | expr ASSIGN_OCTUP expr { Assign($1, Binop($1,Octup,$3) )}
  | expr ASSIGN_OCTDOWN expr { Assign($1, Binop($1,Octdown,$3) )}
  | expr ASSIGN_CONCAT expr { Assign($1, Binop($1,Concat,$3) )}
  | expr ASSIGN_INC expr { Assign($1, Binop($1,IncDuration,$3) )}
  | expr ASSIGN_DEC expr { Assign($1, Binop($1,DecDuration,$3) )}
  | expr STRETCH expr      { Binop($1, Stretch, $3) }
  | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
  | LPAREN expr RPAREN { $2 }
  | LBRACKET actuals_opt RBRACKET { Array($2) }
  | ID index_opt { Index($1,$2) }

/* passed in args to a function */
actuals_opt:
       /* nothing */ { [] }
  | actuals_list  { List.rev $1 }

actuals_list:
       expr                        { [$1] }
  | actuals_list COMMA expr { $3 :: $1 }

index_opt:
  indices { List.rev $1 }

indices:
  LBRACKET LITERAL RBRACKET { [$2] }
  | indices LBRACKET LITERAL RBRACKET {$3 :: $1}
```

8.1.3 ast.ml

```
type op = Plus | Minus | Longer | Shorter | Equal | Neq | Concat | Octup | Less | Leq
| Greater | Geq | Octdown | Halfup | Halfdown | IncDuration | DecDuration | Stretch

type expr =
    Literal of int
  | Note of string
  | Rest of string
  | Id of string
  | Array of expr list
  | Binop of expr * op * expr
  | Assign of expr * expr
  | Call of string * expr list
  | Index of string * expr list
```

```
type stmt =
    Block of stmt list
  | Expr of expr
  | Return of expr
  | If of expr * stmt * stmt
  | While of expr * stmt
  | Loop of expr * expr * stmt

type func_decl = {
       fname : string;
       formals : string list;
       locals : string list;
       body : stmt list;
  }

type program = string list * func_decl list

(*list to string*)
let string_of_list list_t =
let string_temp = String.concat ", " list_t in
"[" ^ string_temp ^ "]";;


let rec string_of_expr = function
       Literal(l) -> string_of_int l
  | Id(i) -> i
  | Note(n) -> n
  | Rest(r) -> r
  | Index(i,a) -> i
  | Binop(e1, o, e2) ->
       string_of_expr e1 ^ " " ^
       (match o with
  Plus -> "+" | Minus -> "-" | Longer -> "*" | Shorter -> "/"
       | Equal -> "==" | Neq -> "!="
       | Less -> "<" | Leq -> "<=" | Greater -> ">" | Geq -> ">="
       | Concat -> "::" | Halfup -> "++" | Halfdown -> "--"
       | Octup -> ">>" | Octdown -> "<<" | IncDuration -> ".*"
       | Stretch -> "->" | DecDuration -> "./") ^ " " ^
       string_of_expr e2
  | Assign(v, e) -> string_of_expr v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
       f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Array(a) -> "[" ^ build a ^ "]" and build = function
                     hd :: [] -> (string_of_expr hd)
                     | hd :: tl -> ((string_of_expr hd) ^ "," ^ (build tl))
                     | _ -> "some thing wrong"

let rec string_of_stmt = function
       Block(stmts) ->
       "{\n" ^ String.concat "" (List.map string_of_stmt stmts) ^ "}\n"
  | Expr(expr) -> string_of_expr expr ^ ";\n";
  | Return(expr) -> "return " ^ string_of_expr expr ^ ";\n";
  | If(e, s, Block([])) -> "if (" ^ string_of_expr e ^ ")\n" ^ string_of_stmt s
  | If(e, s1, s2) ->  "if (" ^ string_of_expr e ^ ")\n" ^
```

```
       string_of_stmt s1 ^ "else\n" ^ string_of_stmt s2
  | While(e, s) -> "while (" ^ string_of_expr e ^ ") " ^ string_of_stmt s
  | Loop(e1, e2, s) -> "loop (" ^ string_of_expr e1 ^ ":" ^  string_of_expr e2 ^
")\n" ^ string_of_stmt s

let string_of_vdecl id = "int " ^ id ^ ";\n"

let string_of_fdecl fdecl =
  fdecl.fname ^ "(" ^ String.concat ", " fdecl.formals ^ ")\n{\n" ^
  String.concat "" (List.map string_of_stmt fdecl.body) ^
  "}\n"

let string_of_program (vars, funcs) =
  String.concat "" (List.map string_of_vdecl vars) ^ "\n" ^
  String.concat "\n" (List.map string_of_fdecl funcs)
```

8.1.4 datalib.ml

```
module NameMap = Map.Make(struct
  type t = string
  let compare x y = Pervasives.compare x y
end)

module IntMap = Map.Make(struct
  type t = int
  let compare x y = Pervasives.compare x y
end)

let maxNoteInt = 120;;
let minNoteInt = 0;;

let noteToIntMap = NameMap.empty;;
let noteToIntMap = NameMap.add "A" 0 noteToIntMap;;
let noteToIntMap = NameMap.add "A#" 1 noteToIntMap;;
let noteToIntMap = NameMap.add "Bb" 1 noteToIntMap;;
let noteToIntMap = NameMap.add "B" 2 noteToIntMap;;
let noteToIntMap = NameMap.add "Cb" 2 noteToIntMap;;
let noteToIntMap = NameMap.add "B#" 3 noteToIntMap;;
let noteToIntMap = NameMap.add "C" 3 noteToIntMap;;
let noteToIntMap = NameMap.add "C#" 4 noteToIntMap;;
let noteToIntMap = NameMap.add "Db" 4 noteToIntMap;;
let noteToIntMap = NameMap.add "D" 5 noteToIntMap;;
let noteToIntMap = NameMap.add "D#" 6 noteToIntMap;;
let noteToIntMap = NameMap.add "Eb" 6 noteToIntMap;;
let noteToIntMap = NameMap.add "E" 7 noteToIntMap;;
let noteToIntMap = NameMap.add "Fb" 7 noteToIntMap;;
let noteToIntMap = NameMap.add "E#" 8 noteToIntMap;;
let noteToIntMap = NameMap.add "F" 8 noteToIntMap;;
let noteToIntMap = NameMap.add "F#" 9 noteToIntMap;;
let noteToIntMap = NameMap.add "Gb" 9 noteToIntMap;;
let noteToIntMap = NameMap.add "G" 10 noteToIntMap;;
let noteToIntMap = NameMap.add "G#" 11 noteToIntMap;;
let intToNoteMap = IntMap.empty;;
let intToNoteMap = IntMap.add 0 "A" intToNoteMap;;
let intToNoteMap = IntMap.add 1 "A#" intToNoteMap;;
let intToNoteMap = IntMap.add 2 "B" intToNoteMap;;
let intToNoteMap = IntMap.add 3 "C" intToNoteMap;;
let intToNoteMap = IntMap.add 4 "C#" intToNoteMap;;
let intToNoteMap = IntMap.add 5 "D" intToNoteMap;;
let intToNoteMap = IntMap.add 6 "D#" intToNoteMap;;
let intToNoteMap = IntMap.add 7 "E" intToNoteMap;;
let intToNoteMap = IntMap.add 8 "F" intToNoteMap;;
let intToNoteMap = IntMap.add 9 "F#" intToNoteMap;;
let intToNoteMap = IntMap.add 10 "G" intToNoteMap;;
let intToNoteMap = IntMap.add 11 "G#" intToNoteMap;;

let noteToDuration = fun x ->
    if String.contains x '.' then
         int_of_string(String.sub x ((String.index x '.')+1) ((String.length x) -
((String.index x '.')+1)))
    else 4

let extractNoteWithoutDuration = fun x ->
    if String.contains x '.' then
         String.sub x 0 (String.index x '.')
    else x

let setNoteDuration = fun x y ->
      (extractNoteWithoutDuration x) ^ "." ^ (string_of_int y)

let noteToInt = fun x ->
    let octave = String.get x ((String.length x)-1) in
        let basicNote = String.sub x 0 ((String.length x)-1) in
        ((NameMap.find basicNote noteToIntMap) + ((int_of_char octave)-48) * 12)

let intToNote = fun x ->
    if x > maxNoteInt then raise (Failure ("Note higher than allowable threshold"))
      else if x < minNoteInt then raise (Failure ("Note lower than allowable
threshold"))
      else (IntMap.find (x - 12*(x / 12)) intToNoteMap) ^ (string_of_int (x / 12))
```

8.1.5 compile.ml

```
open Ast
open Printf
open Datalib

module NameMap = Map.Make(struct
  type t = string
  let compare x y = Pervasives.compare x y
end)

exception ReturnException of expr * expr NameMap.t
```

```
let trackCounters = ref NameMap.empty
let currentTrack = ref ""

let selectTrack v = let oc =  open_out_gen [Open_creat; Open_append; Open_text] 0o666
"output.rym" in
       fprintf oc "\nTrack: %s\n" v;
       close_out oc;
       currentTrack := v;
       ;Literal(0), (NameMap.empty, NameMap.empty)

let writeToFile v = let oc = open_out_gen [Open_creat; Open_append; Open_text] 0o666
"output.rym" in
  let startingTrackCount = if NameMap.mem !currentTrack !trackCounters then
NameMap.find !currentTrack !trackCounters else 0 in

  let printPrim startTick = function
   Note(n) -> let rec write start stop =
       if (start < stop) then
       (fprintf oc "%s %s\n" (string_of_int (startingTrackCount + start))
(string_of_int (noteToInt (extractNoteWithoutDuration n)));
       write (start + 1) stop)
       else stop
       in
       write startTick (startTick + (16 / (noteToDuration n)))
   | Rest(r) -> (startTick + (16 / (noteToDuration r)))
   | _ -> raise (Failure ("Illegal array value"))
  in

  let rec printChord largest startTick = function
       [] -> raise (Failure ("Cannot print empty array"))
       | hd :: [] -> let length = (match hd with
       Array(a) -> printChord largest startTick a
       | Rest(r) -> printPrim startTick (Rest(r))
       | Note(n) -> printPrim startTick (Note(n))
       | _ -> raise (Failure ("Illegal array value"))) in if largest >= length then
largest else length
       | hd :: tl -> let length = (match hd with
       Array(a) -> printChord largest startTick a
       | Rest(r) -> printPrim startTick (Rest(r))
       | Note(n) -> printPrim startTick (Note(n))
       | _ -> raise (Failure ("Illegal array value"))) in if largest >= length then
printChord largest startTick tl
                       else printChord length startTick tl
       in

  let rec printTrack startTick = function
       [] -> raise (Failure ("Cannot print empty array"))
       | hd :: [] -> (match hd with
       Array(a) -> printChord 0 startTick a
       | Rest(r) -> printPrim startTick (Rest(r))
       | Note(n) -> printPrim startTick (Note(n))
       | _ -> raise (Failure ("Illegal array value")))
```

```
        | hd :: tl -> let newStartTick =
        (match hd with
        Array(a) -> printChord 0 startTick a
        | Rest(r) -> printPrim startTick (Rest(r))
        | Note(n) -> printPrim startTick (Note(n))
        | _ -> raise (Failure ("Illegal array value")))
        in printTrack newStartTick tl
        in
   let parse = function
    Note(n) -> let numTicks = printPrim 0 (Note(n)) in trackCounters := NameMap.add
 !currentTrack (startingTrackCount + numTicks) !trackCounters
    | Rest(r) -> let numTicks = printPrim 0 (Rest(r)) in trackCounters := NameMap.add
 !currentTrack (startingTrackCount + numTicks) !trackCounters
    | Array(a) -> let numTicks = printTrack 0 a in trackCounters := NameMap.add
 !currentTrack (startingTrackCount + numTicks) !trackCounters
    | _ -> raise (Failure ("Cannot print anything except arrays, notes, or rests"))
   in
   parse v;
   print_endline "RYM file created successfully";
   close_out oc;
   Literal(0), (NameMap.empty, NameMap.empty)




(* Main entry point: run a program *)
let run (vars, funcs) =
  (* Put function declarations in a symbol table *)
   let func_decls = List.fold_left
        (fun funcs fdecl -> NameMap.add fdecl.fname fdecl funcs)
        NameMap.empty funcs
   in

  (* Invoke a function and return an updated global symbol table *)
   let rec call fdecl actuals globals =



 (* Evaluate an expression and return (value, updated environment) *)
        let rec eval env = function
 Literal(i) -> Literal(i), env
        (*| Noexpr -> 1, env (* must be non-zero for the for loop predicate *)*)
    | Note(n) -> Note(n), env
    | Rest(n) -> Rest(n), env
    | Array(n) -> Array(n), env
    | Index(a,i) -> let v, (locals, globals) = eval env (Id(a)) in
        let rec lookup arr indices =
        let arr = match arr with
        Array(x) -> x
        | _ -> raise (Failure ("Attempting to index a variable that isn't an array"))
        in
        match indices with
        [] -> raise (Failure ("Cannot index empty list"))
```

```
          | Literal(hd) :: [] -> List.nth arr hd, env
          | Literal(hd) :: tl -> lookup (List.nth arr hd) tl
          | Id(hd) :: [] -> let hd,v = eval env (Id(hd)) in (match hd with
                       Literal(hd) -> List.nth arr hd, env

          | _ -> raise (Failure "Illegal index"))
          | Id(hd) :: tl -> let hd,v = eval env (Id(hd)) in (match hd with
                               Literal(hd) -> lookup (List.nth arr hd) tl
          | _ -> raise (Failure "Illegal index"))
          | _ -> raise (Failure "Illegal index")

        in
        lookup v i
        | Binop(e1, op, e2) ->
        let v1, (locals, globals) = eval env e1 in
        let v2, (locals, globals) = eval env e2 in
        let boolean i = if i then 1 else 0 in
        (*let locals, globals = env in*)
        let rec goThroughArray op e l =
        (match op with
        Plus -> (match e with
        Note(n) -> (match l with
               [] -> raise (Failure ("Cannot perform operation on an empty array"))
               | hd :: [] -> (match hd with
               Array(a2) -> [Array((goThroughArray op (Note(n)) a2))]
               | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Note(n)) [v]
               | Index(a,i) -> let v, vars = eval env (Index(a,i)) in goThroughArray op
(Note(n)) [v]
               | Note(n2) -> [Array([Note(n2);Note(n)])]
               | Rest(r) -> raise (Failure ("Cannot make chords with rests"))
               | _  as extra -> raise (Failure ("Illegal array value: " ^
(string_of_expr extra))))
               | hd :: tl -> (match hd with
               Array(a2) -> [Array((goThroughArray op (Note(n)) a2))] @ (goThroughArray
op (Note(n)) tl)
               | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Note(n)) [v] @ (goThroughArray op (Note(n)) tl)
               | Index(a,i) -> let v, vars = eval env (Index(a,i)) in (goThroughArray
op (Note(n)) [v]) @ (goThroughArray op (Note(n)) tl)
               | Note(n2) -> [Array([Note(n2);Note(n)])] @ (goThroughArray op (Note(n))
tl)
               | Rest(r) -> raise (Failure ("Cannot make chords with rests"))
               | _  as extra -> raise (Failure ("Illegal array value: " ^
(string_of_expr extra)))))
        | Literal(lit) -> (match l with
               [] -> raise (Failure ("Cannot perform operation on an empty array"))
               | hd :: [] -> (match hd with
               Array(a2) -> [Array((goThroughArray op (Literal(lit)) a2))]
               | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Literal(lit)) [v]
               | Index(a,i) -> let v, vars = eval env (Index(a,i)) in goThroughArray op
(Literal(lit)) [v]
               | Rest(r) -> [Rest(r)]
```

```
                | Note(n2) ->  let dur = noteToDuration n2 in
                        let oldNote = extractNoteWithoutDuration n2 in
                        [Note((setNoteDuration (intToNote ((noteToInt oldNote) + lit)))
dur)]
                | _  as extra -> raise (Failure ("Illegal array value: " ^
(string_of_expr extra))))
                | hd :: tl -> (match hd with
                Array(a2) -> [Array((goThroughArray op (Literal(lit)) a2))] @
(goThroughArray op (Literal(lit)) tl)
                | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Literal(lit)) [v] @ (goThroughArray op (Literal(lit)) tl)
                | Index(a,i) -> let v, vars = eval env (Index(a,i)) in (goThroughArray
op (Literal(lit)) [v]) @ (goThroughArray op (Literal(lit)) tl)
                | Rest(r) -> [Rest(r)]
                | Note(n2) ->  let dur = noteToDuration n2 in
                        let oldNote = extractNoteWithoutDuration n2 in
                        [Note((setNoteDuration (intToNote ((noteToInt oldNote) + lit)))
dur)]
                        @ (goThroughArray op (Literal(lit)) tl)
                | _  as extra -> raise (Failure ("Illegal array value: " ^
(string_of_expr extra)))))
        | _ -> raise (Failure ("Unuseable value")))
        | Minus -> (match e with
             Literal(lit) -> (match l with
                [] -> raise (Failure ("Cannot perform operation on an empty array"))
                | hd :: [] -> (match hd with
                Array(a2) -> [Array((goThroughArray op (Literal(lit)) a2))]
                | Rest(r) -> [Rest(r)]
                | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Literal(lit)) [v]
                | Index(a,i) -> let v, vars = eval env (Index(a,i)) in goThroughArray op
(Literal(lit)) [v]
                | Note(n2) ->  let dur = noteToDuration n2 in
                        let oldNote = extractNoteWithoutDuration n2 in
                        [Note((setNoteDuration (intToNote ((noteToInt oldNote) - lit)))
dur)]
                | _ -> raise (Failure ("Illegal array value")))
                | hd :: tl -> (match hd with
                Array(a2) -> [Array((goThroughArray op (Literal(lit)) a2))] @
(goThroughArray op (Literal(lit)) tl)
                | Rest(r) -> [Rest(r)]
                | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Literal(lit)) [v] @ (goThroughArray op (Literal(lit)) tl)
                | Index(a,i) -> let v, vars = eval env (Index(a,i)) in (goThroughArray
op (Literal(lit)) [v]) @ (goThroughArray op (Literal(lit)) tl)
                | Note(n2) ->  let dur = noteToDuration n2 in
                        let oldNote = extractNoteWithoutDuration n2 in
                        [Note((setNoteDuration (intToNote ((noteToInt oldNote) - lit)))
dur)]
                        @ (goThroughArray op (Literal(lit)) tl)
                | _ -> raise (Failure ("Illegal array value"))))
        | _ -> raise (Failure ("Unuseable value")))
        | IncDuration -> (match e with
        Literal(lit) -> (match l with
              [] -> raise (Failure ("Cannot perform operation on an empty array"))
              | hd :: [] -> (match hd with
              Array(a2) -> [Array((goThroughArray op (Literal(lit)) a2))]
              | Note(n2) -> let newDuration = (noteToDuration n2) * lit in
                     let newNote = setNoteDuration n2 newDuration in
                     [Note(newNote)]
              | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Literal(lit)) [v]
              | Index(a,i) -> let v, vars = eval env (Index(a,i)) in goThroughArray op
(Literal(lit)) [v]
              | Rest(r) -> let newDuration = (noteToDuration r) * lit in
                     let newRest = setNoteDuration r newDuration in
                     [Rest(newRest)]
              | _ -> raise (Failure ("Illegal array value")))
              | hd :: tl -> (match hd with
              Array(a2) -> [Array((goThroughArray op (Literal(lit)) a2))] @
(goThroughArray op (Literal(lit)) tl)
              | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Literal(lit)) [v] @ (goThroughArray op (Literal(lit)) tl)
              | Index(a,i) -> let v, vars = eval env (Index(a,i)) in (goThroughArray
op (Literal(lit)) [v]) @ (goThroughArray op (Literal(lit)) tl)
              | Note(n2) -> let newDuration = (noteToDuration n2) * lit in
                     let newNote = setNoteDuration n2 newDuration in
                     [Note(newNote)] @ (goThroughArray op (Literal(lit)) tl)
              | Rest(r) -> let newDuration = (noteToDuration r) * lit in
                     let newRest = setNoteDuration r newDuration in
                     [Rest(newRest)] @ (goThroughArray op (Literal(lit)) tl)
              | _ -> raise (Failure ("Illegal array value"))))
       | _ -> raise (Failure ("Unuseable value")))
       | DecDuration -> (match e with
       Literal(lit) -> (match l with
              [] -> raise (Failure ("Cannot perform operation on an empty array"))
              | hd :: [] -> (match hd with
              Array(a2) -> [Array((goThroughArray op (Literal(lit)) a2))]
              | Note(n2) -> let newDuration = (noteToDuration n2) / lit in
                     let newNote = setNoteDuration n2 newDuration in
                     [Note(newNote)]
              | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Literal(lit)) [v]
              | Index(a,i) -> let v, vars = eval env (Index(a,i)) in goThroughArray op
(Literal(lit)) [v]
              | Rest(r) -> let newDuration = (noteToDuration r) / lit in
                     let newRest = setNoteDuration r newDuration in
                     [Rest(newRest)]
              | _ -> raise (Failure ("Illegal array value")))
              | hd :: tl -> (match hd with
              Array(a2) -> [Array((goThroughArray op (Literal(lit)) a2))] @
(goThroughArray op (Literal(lit)) tl)
              | Binop(e1, o, e2) -> let v, vars = eval env (Binop(e1, o, e2)) in
goThroughArray op (Literal(lit)) [v] @ (goThroughArray op (Literal(lit)) tl)
              | Index(a,i) -> let v, vars = eval env (Index(a,i)) in (goThroughArray
op (Literal(lit)) [v]) @ (goThroughArray op (Literal(lit)) tl)
              | Note(n2) -> let newDuration = (noteToDuration n2) / lit in
                     let newNote = setNoteDuration n2 newDuration in
```

```
                    [Note(newNote)] @ (goThroughArray op (Literal(lit)) tl)
               | Rest(r) -> let newDuration = (noteToDuration r) / lit in
                    let newRest = setNoteDuration r newDuration in
                    [Rest(newRest)] @ (goThroughArray op (Literal(lit)) tl)
               | _ -> raise (Failure ("Illegal array value"))))
       | _ -> raise (Failure ("Unuseable value")))
       | _ -> raise (Failure ("Illegal operation")))
       in
       (match op with
       Plus -> (match (v1,v2) with
       (Note(n1),Note(n2)) -> Array([Note(n1);Note(n2)]), env
       | (Literal(l1), Literal(l2)) -> Literal(l1+l2), env
       | (Note(n1),Literal(l2)) -> let dur = noteToDuration n1 in
             let oldNote = extractNoteWithoutDuration n1 in
             Note((setNoteDuration (intToNote ((noteToInt oldNote) + l2))) dur), env
       | (Literal(l1),Note(n2)) -> let dur = noteToDuration n2 in
             let oldNote = extractNoteWithoutDuration n2 in
              Note((setNoteDuration (intToNote ((noteToInt oldNote) + l1))) dur), env
       | (Array(a),Note(n)) -> Array(goThroughArray Plus (Note(n)) a), env
       | (Note(n),Array(a)) -> Array(goThroughArray Plus (Note(n)) a), env
       | (Array(a),Literal(l)) -> Array(goThroughArray Plus (Literal(l)) a), env
       | (Literal(l),Array(a)) -> Array(goThroughArray Plus (Literal(l)) a), env
       | (Rest(r),Literal(l)) -> Rest(r), env
       | (Literal(l),Rest(r)) -> Rest(r), env
       | (Array(a1),Array(a2)) ->  let split big small =
             let arrBig = Array.of_list big
             in
             let working, leftovers =
               Array.to_list (Array.sub arrBig 0 (List.length small)),
               Array.to_list (Array.sub arrBig (List.length small) ((List.length
big) - (List.length small)))
             in
               Array((List.map (fun (x,y) -> Array([x;y])) (List.combine working
small)) @ leftovers)
             in
             if List.length a1 > List.length a2 then
             (split a1 a2), env
             else if List.length a2 > List.length a1 then
             (split a2 a1), env
             else
               Array(List.map (fun (x,y) -> Array([x;y])) (List.combine a1 a2)),env
         | _ -> raise (Failure ("Invalid Plus Operation")))
       |Minus ->
       (match (v1,v2) with
       (Literal(l1), Literal(l2)) -> Literal(l1-l2), env
       | (Note(n1),Literal(l2)) -> let dur = noteToDuration n1 in
             let oldNote = extractNoteWithoutDuration n1 in
             Note((setNoteDuration (intToNote ((noteToInt oldNote) - l2))) dur), env
       | (Literal(l1),Note(n2)) -> let dur = noteToDuration n2 in
             let oldNote = extractNoteWithoutDuration n2 in
             Note((setNoteDuration (intToNote ((noteToInt oldNote) - l1))) dur), env
       | (Array(a),Literal(l)) -> Array(goThroughArray Minus (Literal(l)) a), env
       | (Literal(l),Array(a)) -> Array(goThroughArray Minus (Literal(l)) a), env
       | (Rest(r),Literal(l)) -> raise (Failure ("Cannot change the pitch of a rest"))
       | (Literal(l),Rest(r)) -> raise (Failure ("Cannot change the pitch of a rest"))
       | _ -> raise (Failure ("Invalid Minus Operation")))
       |Equal ->
       (match (v1,v2) with
       (Literal(l1), Literal(l2)) -> Literal(boolean (v1 = v2)), env
       | (Note(n1), Note(n2)) -> let isSame = ((extractNoteWithoutDuration n1) =
(extractNoteWithoutDuration n2)) && ((noteToDuration n1) = (noteToDuration n2))
         in Literal((boolean isSame)), env
       | (Rest(r1), Rest(r2)) -> Literal(boolean (r1 = r2)), env
       | (Array(a1), Array(a2)) -> Literal(boolean (a1 = a2)), env
       | _ -> Literal(0), env)
       |Neq ->
       (match (v1,v2) with
       (Literal(l1), Literal(l2)) -> Literal(boolean (v1 <> v2)), env
       | (Note(n1), Note(n2)) -> let isSame = ((extractNoteWithoutDuration n1) <>
(extractNoteWithoutDuration n2)) || ((noteToDuration n1) <> (noteToDuration n2))
        in Literal((boolean isSame)), env
       | (Rest(r1), Rest(r2)) -> Literal(boolean (r1 <> r2)), env
       | (Array(a1), Array(a2)) -> Literal(boolean (a1 <> a2)), env
       | _ -> Literal(1), env)
       |Less ->
       (match (v1,v2) with
       (Literal(l1), Literal(l2)) -> Literal(boolean (v1 < v2)), env
       | (Note(n1), Note(n2)) -> Literal(boolean ((noteToInt n1) < (noteToInt n2))),
env
       | _ -> raise (Failure ("Invalid Less Than Operation")))
       |Leq ->
       (match (v1,v2) with
       (Literal(l1), Literal(l2)) -> Literal(boolean (v1 <= v2)), env
       | (Note(n1), Note(n2)) -> Literal(boolean ((noteToInt n1) <= (noteToInt n2))),
env
       | _ -> raise (Failure ("Invalid Less Than Or Equal Operation")))
       |Greater ->
       (match (v1,v2) with
       (Literal(l1), Literal(l2)) -> Literal(boolean (v1 > v2)), env
       | (Note(n1), Note(n2)) -> Literal(boolean ((noteToInt n1) > (noteToInt n2))),
env
       | _ -> raise (Failure ("Invalid Greater Than Operation")))
       |Geq ->
       (match (v1,v2) with
       (Literal(l1), Literal(l2)) -> Literal(boolean (v1 >= v2)), env
       | (Note(n1), Note(n2)) -> Literal(boolean ((noteToInt n1) >= (noteToInt n2))),
env
       | _ -> raise (Failure ("Invalid Greater Than Or Equal Operation")))
       |Concat->
       (match (v1,v2) with
       (Array(a1),Array(a2)) ->  Array(a1 @ a2), env
       | (Array(a1), Note(n1)) -> Array(a1 @ [Note(n1)]), env
       | (Array(a1), Rest(r1)) -> Array(a1 @ [Rest(r1)]), env
       | (Rest(r1), Array(a1)) -> Array([Rest(r1)] @ a1), env
       | (Note(n1), Array(a1)) -> Array([Note(n1)] @ a1), env
       | (Note(n1),Note(n2)) -> Array([Note(n1);Note(n2)]), env
       | (Rest(r1),Rest(r2)) -> Array([Rest(r1);Rest(r2)]), env
       | _ -> raise (Failure ("Invalid Concatenation Operation")))
```

```
        |Octup ->
        (match v2 with
        Literal(l) -> eval env (Binop(v1, Plus, Literal(12*l)))
        | _ -> raise (Failure ("LHS of octave shift must be a note or array, RHS must
be an integer")))
        |Octdown ->
        (match v2 with
        Literal(l) -> eval env (Binop(v1, Minus, Literal(12*l)))
        | _ -> raise (Failure ("LHS of octave shift must be a note or array, RHS must
be an integer")))
        |DecDuration ->
        (match (v1,v2) with
        (Note(n1), Literal(l2)) -> let dur = (noteToDuration n1) in
        if dur = 16 then raise (Failure ("Sixteenth notes cannot be shortened"))
        else let newDuration = dur * l2 in
        let newNote = setNoteDuration n1 newDuration in
        Note(newNote), env
        | (Literal(l1), Note(n2)) -> let dur = (noteToDuration n2) in
        if dur = 16 then raise (Failure ("Sixteenth notes cannot be shortened"))
        else let newDuration = dur * l1 in
        let newNote = setNoteDuration n2 newDuration in
        Note(newNote), env
        | (Rest(r1), Literal(l2)) -> let newDuration = (noteToDuration r1) * l2 in
        let newRest = setNoteDuration r1 newDuration in
        Note(newRest), env
        | (Literal(l1), Rest(r2)) -> let newDuration = (noteToDuration r2) * l1 in
        let newRest = setNoteDuration r2 newDuration in
        Note(newRest), env
        | (Array(a),Literal(l)) -> Array(goThroughArray IncDuration (Literal(l)) a),
env
        | (Literal(l),Array(a)) -> Array(goThroughArray IncDuration (Literal(l)) a),
env
        | _ -> raise (Failure ("Invalid Increase Duration Operation")))
        |IncDuration ->
        (match (v1,v2) with
        (Note(n1), Literal(l2)) -> let dur = (noteToDuration n1) in
        if dur = 1 then raise (Failure ("Whole notes cannot be lengthened"))
        else let newDuration = dur / l2 in
        let newNote = setNoteDuration n1 newDuration in
        Note(newNote), env
        | (Literal(l1), Note(n2)) -> let dur = (noteToDuration n2) in
        if dur = 1 then raise (Failure ("Whole notes cannot be lengthened"))
        else let newDuration = dur / l1 in
        let newNote = setNoteDuration n2 newDuration in
        Note(newNote), env
        | (Rest(r1), Literal(l2)) -> let newDuration = (noteToDuration r1) / l2 in
        let newRest = setNoteDuration r1 newDuration in
        Note(newRest), env
        | (Literal(l1), Rest(r2)) -> let newDuration = (noteToDuration r2) / l1 in
        let newRest = setNoteDuration r2 newDuration in
        Note(newRest), env
        | (Array(a),Literal(l)) -> Array(goThroughArray DecDuration (Literal(l)) a),
env
        | (Literal(l),Array(a)) -> Array(goThroughArray DecDuration (Literal(l)) a),
 env
        | _ -> raise (Failure ("Invalid Decrease Duration Operation")))
        |Stretch ->
        (match (v1,v2) with
        (Array(arr),Literal(l)) -> let rec stretch l a =
               if (l > 0) then stretch (l-1) (arr @ a) else a in
               Array(stretch l []), env
        | (_,Literal(l)) -> let rec stretch l a =
               if (l > 0) then stretch (l-1) (v1 :: a) else a in
               Array(stretch l []), env
        | _ -> raise (Failure ("RHS of stretch statement must be an integer")))
        | _ ->raise (Failure ("Invalid operation")))

        | Id(var) ->
    let locals, globals = env in
    if NameMap.mem var locals then
        (NameMap.find var locals), env
    else if NameMap.mem var globals then
        (NameMap.find var globals), env
    else raise (Failure ("undeclared identifier " ^ var))
        | Assign(var, e) ->
    let v, (locals, globals) = eval env e in
    (match var with
        Id(name) ->
        if NameMap.mem name locals then
        v, (NameMap.add name v locals, globals)
        else if NameMap.mem name globals then
        v, (locals, NameMap.add name v globals)
        else
        v, (NameMap.add name v locals, globals)
        | Index(arrName,indices) ->
        let rec getIndexFromVar = function
        Literal(l) -> l
        | Id(i) -> let e, v = eval env (Id(i)) in getIndexFromVar e
        | _ -> raise (Failure ("Illegal index"))
          in let rec swap exps = function
        [] -> raise (Failure ("Cannot perform operation on an empty array"))
        | hd :: [] -> let hd = getIndexFromVar hd in let arr = (Array.of_list exps) in
 arr.(hd) <- v; Array.to_list arr
        | hd :: tl ->       let hd = getIndexFromVar hd in
        let arr = (Array.of_list exps) in let piece = match arr.(hd) with
        Array(x) -> x
        | _ -> raise (Failure ("Bad index"))
        in arr.(hd) <- Array(swap piece tl); Array.to_list arr
        if (NameMap.mem arrName locals) then
        let eList = (match (NameMap.find arrName locals) with
        Array(a) -> a
        | _ -> raise (Failure ("Variable in memory not an array"))) in
        let newArray = Array(swap eList indices) in
        v, (NameMap.add arrName newArray locals, globals)
        else if (NameMap.mem arrName globals) then
        let eList = (match (NameMap.find arrName globals) with
        Array(a) -> a
        | _ -> raise (Failure ("Variable in memory not an array"))) in
```

```
        let newArray = Array(swap eList indices) in
        v, (locals, NameMap.add arrName newArray globals)
        else
        raise (Failure ("Array variable not found in memory"))
        | _ -> raise (Failure ("Can only assign variables or array segments")))

        | Call("print", [e]) ->
    let v, env = eval env e in
    let rec print = function
        Literal(i) -> string_of_int i
        | Note(n) -> n
        | Rest(r) -> r
        | Id(i) -> let expr, vars = eval env (Id(i)) in print expr
        | Index(i,a) -> let expr, vars = eval env (Index(i,a)) in print expr
        | Binop(x,y,z) -> let expr, vars = eval env (Binop(x,y,z)) in print expr
        | Call(x,y) -> let expr, vars = eval env (Call(x,y)) in print expr
        | Assign(x,y) -> let expr, vars = eval env (Assign(x,y)) in print expr
        | Array(a) -> "[" ^ build a ^ "]" and build = function
        hd :: [] -> (print hd)
        | hd :: tl ->  ((print hd) ^ "," ^ (build tl))
        | _ -> raise (Failure ("Unprintable item"))
in
 print_endline (print v);
  Literal(0), env

        | Call(f, actuals) ->
    let fdecl =
        try NameMap.find f func_decls
        with Not_found -> raise (Failure ("undefined function " ^ f))
    in
    let actuals, env = List.fold_left
        (fun (actuals, env) actual ->
  let v, env = eval env actual in v :: actuals, env)
        ([], env) (List.rev actuals)
    in
    let (locals, globals) = env in
    try
        let globals = call fdecl actuals globals
        in Literal(0), (locals, globals)
    with ReturnException(v, globals) -> v, (locals, globals)
        in

        (* Execute a statement and return an updated environment *)
        let rec exec env = function
Block(stmts) -> List.fold_left exec env stmts
        | Expr(e) -> let _, env = eval env e in env
        | If(e, s1, s2) ->
    let v, env = eval env e in
    (match v with
        Literal(l) ->  exec env (if l != 0 then s1 else s2)
        | _ -> raise (Failure ("Error: If only support arithmetic")))
        | While(e, s) ->
    let rec loop env =
        let v, env = eval env e in
```

```
       (match v with
       Literal(l) -> if (l != 0) then loop (exec env s) else env
       | _ -> raise (Failure ("Error: While only support arithmetic")))
    in loop env
    | Loop(e1, e2, s) ->
       let v1, env = eval env e1 in
       let v2, env = eval env e2 in
       (match (v1,v2) with
    (Literal(l1),Literal(l2)) ->
       let rec loop l1 l2 env =
       if (l1 < l2) then loop (l1+1) l2 (exec env s)
       else env
       in
       loop l1 l2 env
       | _ -> raise (Failure ("Both loop parameters must be integers")))
    | Return(e) ->
    let v, (locals, globals) = eval env e in
    raise (ReturnException(v, globals))
       in

       (* Enter the function: bind actual values to formal arguments *)
       let locals =
       try List.fold_left2
    (fun locals formal actual -> NameMap.add formal actual locals)
    NameMap.empty fdecl.formals actuals
       with Invalid_argument(_) ->
  raise (Failure ("wrong number of arguments passed to " ^ fdecl.fname))
       in
       (* Initialize local variables to 0 *)
       let locals = List.fold_left
  (fun locals local -> NameMap.add local (Literal(0)) locals) locals fdecl.locals
       in
       (* Execute each statement in sequence, return updated global symbol table *)
       snd (List.fold_left exec (locals, globals) fdecl.body)

   (* Run a program: initialize global variables to 0, find and run init and then
tracks *)
   in let globals = List.fold_left
       (fun globals vdecl -> NameMap.add vdecl (Literal(0)) globals) NameMap.empty
vars
   in
    let globals = if (NameMap.mem "init" func_decls) then call (NameMap.find "init"
func_decls) [] globals
       else List.fold_left (fun globals vdecl -> NameMap.add vdecl (Literal(0))
globals) NameMap.empty vars
   in NameMap.iter (fun k v ->
       let sub = try
       (String.sub k 0 6)
       with Invalid_argument x -> "notatrack" (* Ignoring *)
       in
       if (String.compare sub "track_") == 0 then
       let trackLabel = (String.sub k 6 ((String.length k) - 6)) in
       try
              ignore (call (NameMap.find k func_decls) [] globals);
              raise (Failure ("return call not made in track file (required)"))
              with ReturnException(v, globals) -> ignore (selectTrack(trackLabel));
ignore (writeToFile(v)); ()
       else () ) func_decls
```

8.1.6 rhythm.ml

```
let _ =
  let lexbuf = Lexing.from_channel stdin in
  let program = Parser.program Scanner.token lexbuf in
       Compile.run program
```

8.1.7 rym2MIDI.java

```
/**
 * CSV2MIDI.java
 * June 11, 2003
 * @author: Stephen Steffes
 * Purpose:  Converts a .csv file to a MIDI file according to ExampleMIDI.csv
 */


import java.io.*;

import javax.sound.midi.*;

import java.lang.*;


public class rym2MIDI{



 public static void main(String[] args) throws InvalidMidiDataException, IOException
{

  int timingRes = 4, instrument = 1;
  //***** Get Inputs *****
  if((args.length<2)||(args.length>4)){
   printUsageAndExit();
  }
  else if(args.length==2){
   instrument = 1;
   timingRes = 4;
  }
  else if(args.length==3){
   String insStr = args[2];
   instrument = Integer.parseInt(insStr.trim());
   timingRes = 4;

  }else if(args.length==4){
   String insStr  = args[2];
   String timeStr = args[3];
   instrument = Integer.parseInt(insStr.trim());
   timingRes = Integer.parseInt(timeStr.trim());
  }


  //instrument and timingRes are default.
  //int timingRes=96, instrument = 110;
  int channel=0,note=0,tick=0,velocity=90,column=0;

  int max_tick = 100000;
  int max_track = 100;
  int max_note = 200;

  Sequence sequence = null;
  File outputFile = new File(args[1]);

  //***** Initialize array ******
  int NoteArr[][][] = new int[max_track][][];
  int trackinfo[][] = new int[max_track][2];


  FileInputStream fstream = new FileInputStream(args[0]);
  // Get the object of DataInputStream
  DataInputStream in = new DataInputStream(fstream);
  BufferedReader br = new BufferedReader(new InputStreamReader(in));


  String strLine;
  int LineNum = 0;
  int track_num = -1;
  int total_track_num = 0;
  int track_assigned = -1;
  int ArrTemp[][] = new int[max_tick][2];
  //Read File Line By Line
  while ((strLine = br.readLine()) != null)   {
   // Print the content on the console
   strLine = strLine.trim();
   if (strLine.indexOf("Track")>=0) {
       //track_num = getTrackNum(strLine);
       track_num = track_num+1;
       LineNum = 0;
       ArrTemp = new int[max_tick][2];
   }
   else if(strLine.equals("")||strLine.equals("EOF")){
       //System.out.println (strLine);
       if(track_assigned != track_num){
       NoteArr[track_num] = new int[LineNum][2];
       for(int i = 0; i < LineNum; i++){
       NoteArr[track_num][i] = ArrTemp[i];
       }
       track_assigned = track_num;
       trackinfo[total_track_num][0] = track_num;
       trackinfo[total_track_num][1] = LineNum;
       total_track_num = total_track_num + 1;
```

```
        }
    }
    else{
        //System.out.println (strLine);
        ArrTemp[LineNum] = getTickAndNote(strLine);
        LineNum = LineNum + 1;
    }
  }

  if(track_assigned != track_num){
   NoteArr[track_num] = new int[LineNum][2];
   for(int i = 0; i < LineNum; i++){
       NoteArr[track_num][i] = ArrTemp[i];
   }
   track_assigned = track_num;
   trackinfo[total_track_num][0] = track_num;
   trackinfo[total_track_num][1] = LineNum;
   total_track_num = total_track_num + 1;
  }

  int nChannels = total_track_num;




  //***** Initialize Sequencer *****
  try{
   sequence = new Sequence(Sequence.PPQ, timingRes);   //initialize sequencer with
timingRes
  }catch (InvalidMidiDataException e){
   e.printStackTrace();
   System.exit(1);
  }


  //***** Create tracks and notes *****
  /* Track objects cannot be created by invoking their constructor
       directly. Instead, the Sequence object does the job. So we
         obtain the Track there. This links the Track to the Sequence
       automatically.
   */
  Track track[] = new Track[nChannels];
  for(int i=0;i<nChannels;i++){
   track[i]=sequence.createTrack();                    //create tracks

   ShortMessage sm = new ShortMessage( );
   sm.setMessage(ShortMessage.PROGRAM_CHANGE, i, instrument, 0);  //put in
instrument[i] in this track
   track[i].add(new MidiEvent(sm, 0));
  }


  for(int ti=0; ti<nChannels;ti++){
   int tracknum = trackinfo[ti][0];
```

```
    int notelen = trackinfo[ti][1];
    int OneTrack[][] = NoteArr[tracknum];
    int NoteMatrix[][] = new int[max_note][max_tick];
    int NoteStat[][] = new int[max_note][2];
    channel = ti;
    for(int ni=0; ni<notelen; ni++){
        tick = OneTrack[ni][0];
        note = OneTrack[ni][1];

        NoteStat[note][0] = 1;
        int nowTicknum = NoteStat[note][1];
        NoteMatrix[note][nowTicknum] = tick;
        int newTicknum = nowTicknum+1;
        NoteStat[note][1] = newTicknum;
    }

    for(int i=0; i<max_note; i++){
        if(NoteStat[i][0]==1){
        int note_value = i;
        int note_len = NoteStat[i][1];
        int tick_last = -999;
        int tick_dur = 0;
        int tick_ons = 0;
        for(int j=0; j<note_len;j++){
        int tick_now = NoteMatrix[note_value][j];
        if(j==0){
        tick_ons = tick_now;
        tick_dur = 1;
        tick_last = tick_now;
        }else if(j==note_len-1){
        if(tick_now==(tick_last+1)){
        tick_dur = tick_dur + 1;
        }else{
        tick_ons = tick_now;
        tick_dur = 1;
        }

        int jump = 0;
        for(int k=tick_ons;k<=(tick_ons+tick_dur);k=k+jump){
        int modnum = k%timingRes;
        jump = timingRes - modnum;
        if((k+jump)<(tick_ons+tick_dur)){
        int tick_ons_oneres = k;
        int tick_dur_oneres = jump;

track[channel].add(createNoteOnEvent(note_value,tick_ons_oneres,channel,velocity));

track[channel].add(createNoteOffEvent(note_value,tick_ons_oneres+tick_dur_oneres,chan
nel));
        }else{
        int tick_ons_oneres = k;
        int tick_dur_oneres = tick_ons+tick_dur-k;

track[channel].add(createNoteOnEvent(note_value,tick_ons_oneres,channel,velocity));
track[channel].add(createNoteOffEvent(note_value,tick_ons_oneres+tick_dur_oneres,chan
nel));
        }
        }

        }
        else if(tick_now==(tick_last+1)){
        tick_dur = tick_dur+1;
        tick_last = tick_now;
        }else{
        int jump = 0;
        for(int k=tick_ons;k<=(tick_ons+tick_dur);k=k+jump){
        int modnum = k%timingRes;
        jump = timingRes - modnum;
        if((k+jump)<(tick_ons+tick_dur)){
        int tick_ons_oneres = k;
        int tick_dur_oneres = jump;

track[channel].add(createNoteOnEvent(note_value,tick_ons_oneres,channel,velocity));

track[channel].add(createNoteOffEvent(note_value,tick_ons_oneres+tick_dur_oneres,chan
nel));
        }else{
        int tick_ons_oneres = k;
        int tick_dur_oneres = tick_ons+tick_dur-k;

track[channel].add(createNoteOnEvent(note_value,tick_ons_oneres,channel,velocity));

track[channel].add(createNoteOffEvent(note_value,tick_ons_oneres+tick_dur_oneres,chan
nel));
        }
        }
        tick_ons = tick_now;
        tick_dur = 1;
        tick_last = tick_now;
        }

        }
        }
    }
   }

   try{
    MidiSystem.write(sequence, 1, outputFile);
   }catch (IOException e){
    e.printStackTrace();
    System.exit(1);
   }
  }
  //turns note on
  private static MidiEvent createNoteOnEvent(int nKey, long lTick,int channel,int
 velocity){
   return createNoteEvent(ShortMessage.NOTE_ON,nKey,velocity,lTick,channel);
  }

  //turns note off
  private static MidiEvent createNoteOffEvent(int nKey, long lTick,int channel){
   return createNoteEvent(ShortMessage.NOTE_OFF,nKey,0,lTick,channel);  //set note to
 0 velocity
  }

  //turns note on or off
  private static MidiEvent createNoteEvent(int nCommand,int nKey,int nVelocity,long
 lTick,int channel){
   ShortMessage message = new ShortMessage();
   try{
    message.setMessage(nCommand,channel,nKey,nVelocity);
   }catch (InvalidMidiDataException e){
    e.printStackTrace();
    System.exit(1);
   }
   MidiEvent event = new MidiEvent(message,lTick);
   return event;
  }

  private static void printUsageAndExit(){
   out("usage:");
   out("java CSV2MIDI <infile.csv> <outfile.midi> <instrument> <timeresolution>");
   System.exit(1);
  }

  private static void out(String strMessage){
   System.out.println(strMessage);
  }

  private static int getTrackNum(String track_label){
   String NumStr="";
   if (track_label.indexOf("\t")>=0){
    int startnum = track_label.indexOf("\t");
    int endnum = track_label.indexOf(":");
    NumStr = track_label.substring(startnum+1,endnum);
   }
   else{
    int startnum = track_label.indexOf("Track");
    int endnum = track_label.indexOf(":");
    NumStr = track_label.substring(startnum+5,endnum);
   }
   int Num = Integer.parseInt(NumStr.trim());
   return Num;
  }
  private static int[] getTickAndNote(String strline){
   int blanknum = strline.indexOf(" ");
   if(blanknum==-1){
    blanknum = strline.indexOf("\t");
   }

   int arr_two[] = new int[2];
   String Tick = strline.substring(0,blanknum);
   String Note = strline.substring(blanknum+1,strline.length());
   arr_two[0] = Integer.parseInt(Tick.trim());
   arr_two[1] = Integer.parseInt(Note.trim());
   return arr_two;

  }

 }
```

8.1.8 rym2ry.m

```
% this function could transfer .rym files back to .ry file
function rym2ry(rymfile)

fid = fopen(rymfile);
tline = fgets(fid);


CellTrack = [];
TrackNum = 0;
while ischar(tline)
       if(strfind(tline,'Track'))
       if TrackNum ~=0
            CellTrack{TrackNum} = ArrayOneTrack;
       end
       TrackNum = TrackNum+1;
        ArrayOneTrack = [];
       count = 1;
       else
       nums = str2num(tline);
       if isempty(nums)

       else
            ArrayOneTrack(count,1) = nums(1);
            ArrayOneTrack(count,2) = nums(2);
              count = count+1;
              fprintf('%d,%d\n',nums(1),nums(2));
       end

       end
       tline = fgets(fid);
end
```

```
firstcount = 0;
n = 1;
for ci = 1:length(CellTrack)
       Array = CellTrack{ci};
       count_min = Array(1,1);
       Array(:,1) = Array(:,1)-count_min+1;
       count_max = Array(end,1);
       Array_str_cell = [];
       for i = 1:count_max
       ind = find(Array(:,1)==i);
       if isempty(ind)
            Array_str_cell{i} = 'rest';
       elseif  length(ind)== 1
              Array_str_cell{i} = int2note(Array(ind,2));
       else
              IntNotes = Array(ind,1);
              strprint = '';
              for ii = 1:length(IntNotes)
              if ii==1
                     strprint = sprintf('[%s',int2note(IntNotes(ii))) ;
              elseif ii == length(IntNotes)
                     strprint = sprintf('%s,%s]',strprint,int2note(IntNotes(ii))) ;
              else
                     strprint = sprintf('%s,%s',strprint,int2note(IntNotes(ii))) ;
              end
              end
            Array_str_cell{i} = strprint;
       end
       end
end
end

function note = int2note(x)
IntToNoteTable = {0,'A';...
       1,'A#';...
       2,'B';...
       3,'C';...
       4,'C#';...
       5,'D';...
       6,'D#';...
       7,'E';...
       8,'F';...
       9,'F#';...
       10,'G';...
       11,'G#'};

num1 = mod(x,12);
num2 = floor(x/12);
note = sprintf('%s%d',IntToNoteTable{num1+1,2},num2);
end
```

8.2 Code of Examples

8.2.1 Row Row Row Your Boat

```
getBaseNotes() {
   def row;
   def rowbase;
   rowbase = [[C5,E5,G5], [C5,E5,G5], [C5,E5,G5],D5.8, E5.8, E5.8, D5.8, E5.8, F5.8,
G5.2, C6, G5, E5, C5, G5.8, F5.8, E5.8, D5.8, [C5,E5,G5]];

  row = rowbase->3;
  return row;
}

track_1() {
   return getBaseNotes();
}

track_2() {
   return R.1->4 :: getBaseNotes() << 2;
}

track_3() {
   return R.1->2 :: getBaseNotes() << 1;
}
```

8.2.2 Shepard Tones

```
track_1() {
 c = [[C5.1,C6.1,C4.1,C3.1,C2.1]];        /* C octaves */
 e = c + 4;                               /* E octaves */
 g = c + 7;                               /* G octaves */
 count = 0;
 song = [];
 while (count < 12) {
song = song :: (c+e+g) :: R.1->2 :: (c+1 + e+1 + g+1) :: R.1->16;
              c++; e++; g++; count++;
       }
 return song->3;
 }
```

8.2.3 Gangnam Style

```
track_1()
{
one1 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4.1];
one2 =
[G#4.1,G#4.2,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.2,G#4.1,G#4.1,G#4.1,B5.1,B5.1,B5.1];
one3 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4,R.2,R];
one4 = [G#4.1,G#4.1,G#4.1,G4.1,G4.1,G4.1,F#4.1,F#4.1,F#4.1,B4.1,B4.1,B4.1];
one5 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4.1];
one6 =
[G#4.1,G#4.2,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.2,G#4.1,G#4.1,G#4.1,B5.1,B5.1,B5.1];
one7 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4,R.2,R];
one8 = [G#4.1,G#4.1,G#4.1,G4.1,G4.1,G4.1,F#4.1,F#4.1,F#4.1,B4.1,B4.1,B4.1];
one9 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4.1];
one10 =
[G#4.1,G#4.2,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.2,G#4.1,G#4.1,G#4.1,B5.1,B5.1,B5.1];
one11 =
[G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4,R.2,R];
one12 = [G#4.1,G#4.1,G#4.1,G4.1,G4.1,G4.1,F#4.1,F#4.1,F#4.1,B4.1,B4.1,B4.1];
one13 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4.1];
one14 =
[G#4.1,G#4.2,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.2,G#4.1,G#4.1,G#4.1,B5.1,B5.1,B5.1];
one15 =
[[C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],R.1,R.2,[
C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],R.1,R.2,[C#
5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1]];
one16 =
[[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],R.1,R.
2,[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],R.1,R
.2,[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1]];
one17 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4.1];
one18 =
[G#4.1,G#4.2,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.2,G#4.1,G#4.1,G#4.1,B5.1,B5.1,B5.1];
one19 =
[G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4,R.2,R];
one20 = [G#4.1,G#4.1,G#4.1,G4.1,G4.1,G4.1,F#4.1,F#4.1,F#4.1,B4.1,B4.1,B4.1];
one21 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4.1];
one22 =
[G#4.1,G#4.2,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.2,G#4.1,G#4.1,G#4.1,B5.1,B5.1,B5.1];
one23 =
[G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4,R.2,R];
one24 = [G#4.1,G#4.1,G#4.1,G4.1,G4.1,G4.1,F#4.1,F#4.1,F#4.1,B4.1,B4.1,B4.1];
one25 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4.1];
one26 =
[G#4.1,G#4.2,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.2,G#4.1,G#4.1,G#4.1,B5.1,B5.1,B5.1];
one27 =
[G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4,R.2,R];
one28 = [G#4.1,G#4.1,G#4.1,G4.1,G4.1,G4.1,F#4.1,F#4.1,F#4.1,B4.1,B4.1,B4.1];
one29 = [G#3.1,G#3.1,G#3.1,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.1,R.1,G#4.1,G#4.1,G#4.1];
one30 =
[G#4.1,G#4.2,G#3.1,G#3.2,G#4.1,G#4.2,R.1,R.2,G#4.1,G#4.1,G#4.1,B5.1,B5.1,B5.1];
one31 =
[[C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],R.1,R.2,[
C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],R.1,R.2,[C#
5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1],[C#5.1,E5.1,G#5.1,C#6.1]];
one32 =
[[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],R.1,R.
2,[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],R.1,R
.2,[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1],[D#5.1,F#5.1,A#6.1,D#6.1]];

onesong =
one1::one2::one3::one4::one9::one10::one11::one12::one13::one14::one15::one16::one17:
:one18::one19::one20::one21::one22::one23::one31::one32;

return onesong;
}

track_2()
{
two1 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two2 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two3 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two4 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two5 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two6 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two7 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two8 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two9 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
```

```
two10 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two11 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two12 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two13 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two14 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two15 = [C#3.1,C#3.1,C#3.1,R.1,R.1,R.1,C#3.1,C#3.1,C#3.1,R.1,R.1,R.1];
two16 = [D#3.1,D#3.1,D#3.1,R.1,R.1,R.1,D#3.1,D#3.1,D#3.1,R.1,R.1,R.1];
two17 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two18 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two19 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two20 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two21 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two22 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two23 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two24 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two25 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two26 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two27 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two28 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two29 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two30 = [G#2.1,G#2.1,G#2.1,R.1,R.1,R.1,G#2.1,G#2.1,G#2.1,R.1,R.1,R.1];
two31 = [C#3.1,C#3.1,C#3.1,R.1,R.1,R.1,C#3.1,C#3.1,C#3.1,R.1,R.1,R.1];
two32 = [D#3.1,D#3.1,D#3.1,R.1,R.1,R.1,D#3.1,D#3.1,D#3.1,R.1,R.1,R.1];

twosong =
two1::two2::two3::two4::two9::two10::two11::two12::two13::two14::two15::two16::two17:
:two18::two19::two20::two21::two22::two23::two31::two32;

return twosong;
}


track_3()
{
three1 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three2 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three3 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three4 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three5 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three6 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three7 = [G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.1,G#5.1,R.1,R.1,R.1,R.1,R.1,R.1];
three8 =
[B6.1,B6.1,B6.1,A#6.2,A#6,A6.2,A6,R.1,R.2,A6.2,A6,G#5.1,G#5.1,G#5.1,G#5.1,G#5.2,R.2,R
];
three9 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three10 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three11 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three12 =
[B6.1,B6.1,B6.1,A#6.2,A#6,A6.2,A6,R.1,R.2,A6.2,A6,G#5.1,G#5.1,G#5.1,G#5.1,G#5.2,R.2,R
];
three13 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three14 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three15 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three16 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three17 =
[C6,C6.8,C#6.1,C#6.8,R.1,R.2,B6.1,B6.2,R.1,R.2,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.1,G#5.1]
;
three18 = [G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R.1,R.1,R.1,R.2];
three19 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.2,R.1,R.2,A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#
5.2,R.1,R.2];
```

```
three20 = [G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R.1,R.1,R.1,R.2,G#5.1,G#5.2,R.1,R.2];
three21 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R.2,A#6,A#6.8,B6.1,B
6.8,R.1,R.2];
three22 =
[G#5.1,G#5.2,R.1,R.2,A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R
.2];
three23 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.2,R.1,R.2,A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#
5.2,R.1,R.2];
three24 = [G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R.1,R.1,R.1,R.2,G#5.1,G#5.2,R.1,R.2];
three25 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.1,G#5.1,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R.
2];
three26 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R
.2];
three27 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.2,R.1,R.2,A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#
5.2,R.1,R.2];
three28 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R.2,G#5.1,G#5.2,R.1,R
.2];
three29 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.1,R.1,R.1,R.2,A#6,A#6.8,B6.1,B6.8,R.1,R.1,R.1,R.1,R.2];
three30 =
[A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#5.2,R.1,R.2,A#6,A#6.8,B6.1,B6.8,R.1,R.2,G#5.1,G#
5.2,R.1,R.2];
three31 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
three32 = [D#6.1,D#6.2,R.1,R.2,D#6.1,D#6.2,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.2];

threesong =
three1::three2::three3::three4::three9::three10::three11::three12::three13::three14::
three15::three16::three17::three18::three19::three20::three21::three22::three23::thre
e31::three32;

return threesong;
}

track_4()
{
four1 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four2 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four3 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four4 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four5 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four6 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four7 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four8 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four9 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four10 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four11 =
[[A3.1,A#4.1],[A3.2,A#4.2],R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four12 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four13 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four14 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four15 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
four16 = [A3.1,A3.2,R.1,R.1,R.1,B3.1,B3.1,B3.1,R.1,R.2,A3.1,A3.2,R.1,R.2];
four17 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four18 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four19 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
```

```
four20 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four21 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four22 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four23 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four24 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four25 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four26 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four27 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four28 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four29 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four30 = [A3.1,A3.2,R.1,R.1,R.1,R.1,R.2,A3.1,A3.2,R.1,R.1,R.1,R.1,R.2];
four31 = [R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1,R.1];
four32 = [A3.1,A3.2,R.1,R.1,R.1,B3.1,B3.1,B3.1,R.1,R.2,A3.1,A3.2,R.1,R.2];

foursong =
four1::four2::four3::four4::four9::four10::four11::four12::four13::four14::four15::fo
ur16::four17::four18::four19::four20::four21::four22::four23::four31::four32;

return foursong;
}
```

8.2.4 arpeggio.ry

```
track_cArp() {
        c = C4;
        song = [];
        loop (0 : 4) {
                song ::= c;
                c += 4;
        }
        song[2]--;
        song ::= song[2] :: song[1] :: song[0];
        return song;
}
```

8.2.5 scale.ry

```
/* major scales are whole, whole, half, whole, whole, whole, half */
/* minor scale: whole half whole whole half whole whole */

cMajor() {
        song = [];
        c = C4;
        steps = [2,2,1,2,2,2,1,1];
        count = 0;
        while(count < 8) {
                song ::= c;
                c += steps[count];
                count++;
        }
        return song;
}

track_cMajor_and_eMinor() {
        song = cMajor() :: R.1->2;
        c = E4;
        steps = [2,1,2,2,1,2,2,2];
        count = 0;
        while(count < 8) {
                song ::= c;
                c += steps[count];
                count++;
        }
        return song;
}
```

8.3 Test Examples

8.3.1 test-add.ry

```
init() {
       print(1+1);
       print(1+2+3);
       print(A4 + 1);
       print(A4 + 1 + 2);
       print([A4,B4] + 1);
       print([A4,B4] + 1 + 2);
       print([A4,B4,[C4,D4]] + 1);
       print(A4 + B4);
       print(A4 + B4 + C4);
       print([A4,B4,C4] + D4);
}
```

8.3.2 test-array.ry

```
init() {

       print([1]);
       print([1,2,3]);
       print([1,2,[1,2,3]]);
}
```

8.3.3 test-concat.ry

```
init() {

       print(A4 :: A4);

}
```

8.3.4 test-equal.ry

```
init() {

       print(A4 == A4);
       print(1 == 1);
       print(A4 == B4);
       print(1 == 2);
       print([A4,B4] == [A4,B4]);
       print([A4,B4,[C4,D4,E4]] == [A4,B4,[C4,D4,E4]]);
       print([A4,B4,[C4,D4,E4]] == [A4,B4,[C4,D4,F4]]);

}
```

8.3.5 test-expr.ry

```
init() {
       print(1 + 2 - 3);
       a = [A4,B4,C4];
       b = A4 - (1 + 2) * 4 / 2 :: [[a[1]+1,B4]] -> 3 >> 3 << 1;
       print(b);
       b++;
       print(b);
       b--;
       print(b);
       b >>= 2;
       print(b);
       b <<= 3;
       print(b);
       b += 1;
       print(b);
       b -= 2;
       print(b);
       b ::= G#7;
       print(b);
       b *= 2;
       print(b);
       b /= 4;
       print(b);
}
```

8.3.6 test-file.ry

```
def s;
```

```
track_foo()
{
  def c;
  c = [[A0.16,A1.16,A2.16],A3,A4.16,R.8,A2];
  return c;
}

track_foo2()
{
  return s;
}

init()
{
  s = [A5,B3,R.1,D7];
}
```

8.3.7 test-function.ry

```
getNotes() {
       return [A4,B4,C4];
}

plusOne(e1) {
       return e1 + 1;
}

add(e1,e2) {
       return e1 + e2;
}



init() {
       a = getNotes();
       b = plusOne(a);
       c = add(a,b);
       print(a);
       print(b);
       print(c);
}
```

8.3.8 test-geq.ry

```
init() {

       print(B4 >= A#4);
       print(B4 >= B4);
       print(B4 >= C4);
}
```

8.3.9 test-global.ry

```
def c;

init() {
       c = [A4,B4,C4];
       testGlobal();
}

testGlobal() {
       print(c);
}
```

8.3.10 test.gt.ry

```
init() {

       print(B4 > A#4);
       print(B4 > B4);
       print(B4 > C4);

}
```

8.3.11 test-if.ry

```
init() {

       if (A4 == A4) {
              print(1);
       }
       else {
              print(2);
       }

       if (A4 != A4) {
              print(3);
       }
       else {
              print(4);
       }

       if (A4 == A4) {
              print(5);
       }
```

```
        if (A4 == A4) {
               if (A4 == A4) {
                      print(6);
               }
        }

        if (A4 == A4) {
               if (A4 != A4) {
                      print(7);
               }
               else {
                      print(8);
               }
        }


}
```

8.3.12 test-index.ry

```
init() {

        a = [A4,B4,C4];
        print(a[1]);
        a = [A4,B4,[C4,D4,E4]];
        print(a[2][1]);

}
```

8.3.13 test-length.ry

```
init() {
        print(A4 * 2);
        print(A4.16 * 2);
        print(A4.2 * 2);
        print(A4.16 * 16);
        print([A4,B4,C4] * 2);
        print([Ab4.8,B4.8,C#4.8] * 2);
}
```

8.3.14 test-leq.ry

```
init() {

        print(B4 <= A#4);
        print(B4 <= B4);
```

```
	print(B4 <= C4);


}
```

8.3.14 test-lit.ry

```
init() {

	print(1);
	print(0);
	print(-1);
	print(100);
	print(-100);

}
```

8.3.15 test-local.ry

```
def c;

foo(c) {
	c--;
	print(c);
	bar();
}

bar() {
	print(c);
}


init() {
	c = C4.4;
	foo(c);
}
```

8.3.16 test-loop.ry

```
init() {

	allNotes = [];
	start = A4;

	loop(0:12) {
		allNotes ::= start;
		start++;
	}

	print(allNotes);

}
```

8.3.17 test-it.ry

```
init() {

	print(B4 < A#4);
	print(B4 < B4);
	print(B4 < C4);

}
```

8.3.18 test-minus.ry

```
init() {
	print(-1 - 1);
	print(1 - 2 - 3);
	print(43 - 7);
	print(43 - -7);
	print(A4 - 1);
	print(A4 - 1 - 2);
	print([C4,F4] - 1);
	print([A4,B4] - 1 - 2);
	print([A4.1,B4.2,[C4.4,D4.8]] - 1);
}
```

8.3.19 test-note.ry

```
init() {

	print(G4);
	print(G#4);
	print(Gb4);
	print(Gb4.4);

}
```

8.3.20 test-octdown.ry

```
init() {
        print(C4 << 2);
}
```

8.3.21 test-octup.ry

```
init() {

        print(C4 >> 2);
}
```

8.3.22 test-rest.ry

```
init() {

        print(R);
        print(R.8);

}
```

8.3.23 test-shorten.ry

```
init() {

        print(A4 / 2);
        print(A4.1 / 16);
        print(A4.1 / 2);
        print(A4.8 / 2);
        print([A4,B4,C4] / 2);
        print([Ab4.8,B4.8,C#4.8] / 2);

}
```

8.3.24 test-stretch.ry

```
init() {

        print(A4 -> 3);
        print([A4,B4] -> 3);
        print([[A4,B4]] -> 3);
}
```

8.3.25 test-unequal.ry

```
init() {

       print(B4 != A#4);
       print(B4 != B4);
       print(B4 != C4);

}
```

8.3.26 test-while.ry

```
init() {

       a = 0;
       while (a<10) {
              print(a);
              a++;
       }

}
```