TENLAB Programming Language Final Report

Mehmet Turkcan, Dallas Jones, Yusuf Cem Subakan
{mkt2126,drj2115,ys2939}@columbia.edu

May 11, 2016

Contents

1 Introduction

1.1 TENLAB:

TENLAB is a MATLAB-like general numerical computation language. TENLAB is designed to be easy to learn and work with, using a natural and streamlined syntax whilst retaining enough versatility to be powerful enough to address the needs of those requiring manipulation of multidimensional arrays for scientific applications. The syntax of TENLAB resembles MATLAB heavily with a few slight changes, but the way tensors are handled are different and control flows operate in a fundamentally different fashion. The TENLAB compiler outputs C code, which can then be compiled into a native binary or a MEX function for use with MATLAB.

1.2 Related Work:

There exist a wide number of languages with different levels of multidimensional array support and distinctive capabilities. In machine learning community, languages like MATLAB [1], R [2] and the ever-popular NumPy+Python [3] workflow as well as libraries like Theano [4], Google's TensorFlow [5] and the recent TensorLab [6] for MATLAB are used to perform a wide variety of domain-specific or broadly general computing tasks. These languages usually depend on either the needs of their authors or the ingenuity and participation of the users and extensive libraries to find their own niche and become successful.

Even though such languages generally thrive thanks to these implicit positive feedback loops, this popularity leads to inescapable bottlenecks at times in which even the most innocuous assumptions lead to difficult situations for the programmer who seeks the environment that offers the shortest amount of steps between the task and what the language offers.

1.3 Goals:

Whereas the syntax of TENLAB does resemble these aforementioned general-purpose computation languages to some extent, the general method of operation for the language is noticably different. Assignments to TENLAB tensors result in elements getting added to the tensors instead. Shape information and the size of the content of tensors are generally independent of each other, and users can reshape and change the values in a tensor at will. This design choice allows for a huge number of possible uses.

1.3.1 Flexibility:

TENLAB allows programmers to alter everything about the internal representation of tensors, allowing paradigms with different internal representations of tensors to be used according to the preferences of the programmers.

1.3.2 Clarity:

TENLAB is a language that seeks to allow the user to do whatever he/she wants without previous planning about the structure of the program. Every variable keeps track of its past by default and the language also contains a number of powerful built-in functions that allow users to control the memory if they want to. TENLAB is also designed to be an easy language which can be learned in minutes.

1.3.3 Symbiosis:

TENLAB effortlessly interfaces with MATLAB via the use of the MEX files, giving users the full capabilities of MATLAB at will. This feature is inspired by the powerful CVX system [7], which is a massively useful tool for addressing a number of difficult optimization problems efficiently. Moreover, as the expected audience for TENLAB consists of individuals who are probably well versed in MATLAB and would rather continue to stay in MATLAB rather than to use two languages concurrently, this aspect of the language is crucial to its future as a viable tool.

2 Tutorial

TENLAB has a clean and descriptive syntax that syntactically resembles MATLAB as suggested by the name; however, control flow statements as well as the general minutiae of the language are slightly different and put the multidimensional nature of the structures the code is manipulating to the forefront.

2.1 Compiling and Running Your First TENLAB Program:

In order to generate executable TENLAB programs, you need to have `gcc` installed and ready to go in your `PATH`. The TENLAB compiler generates C source code by default. Enter the */tenlab_src* folder and type `make` to build the TENLAB-to-C compiler **tenlab**. You can then run **tenlab** directly on your TENLAB source files, which will generate C source files with *.c* extensions automatically. These can then be compiled directly into binaries by calling `gcc`. Or, if you would like to use TENLAB along with MATLAB directly, you can run **tenlab** with the **-m** parameter to generate MEX-ready TENLAB code.

The general structure of a TENLAB program resembles an amalgamation of the function files and scripts of MATLAB that most MATLAB users have secretly desired at one point. The following program demonstrates a number of properties of the language:

Program 1: A Demonstration of TENLAB

```
function triple(X);
  X = X * 3;
  return X;
end;
tensor X;
X = 5;
X=triple(X);
print(X);
%Output:
%5.000000
%15.000000
```

Observe the following:

- A TENLAB program consists of a number of function declarations that follow a script block that operates like a `main` function. Both the functions and the script begin with the declarations of the tensors that exist within their scope followed by the rest of the code.
- The built-in `print` function allows the printing of the contents of tensors, a single element per line.
- Assignments add elements to tensors, rather than replacing their contents.

2.2 Dealing with Tensors:

User-implemented tensors in TENLAB are represented using identifiers. An identifier begins with a letter and continues with a sequence of letters or integers. The only data type in TENLAB is that of `tensor`. Tensors need to be declared at the beginning of the functions or the script block. Values can be assigned to tensors in the following fashion:

```
tensor a;
a = 5;                 % Create a 1D tensor with a single
                       % element
```

```
tensor b;
b = [5];               % Same as the previous assignment
tensor c;
c = [[5,4],[4,3]];     % Create a 2D tensor i.e. a matrix
                       % with elements [5,4;4,3]
```

A very interesting and unique feature in TENLAB is that it allows for repeated matrix products across different dimensions of the tensors using a very terse syntax. Consider the following example:

Program 2: A Gentle Introduction to the Tensor Product

```
tensor X;
tensor Y;
tensor Z;
X = [[3,4],[4,5]];
Y = [[1,2],[3,4]];
Z = [[0,0],[0,0]];
Z = {1} {2,2} X {2} .* {2,2} Y {1};
print(Z);
```

Curly brackets have a very specific meaning in TENLAB. They hold lists of integers separated by `,` characters. The second and the fourth lists give the shape to be used for X and Y respectively during the tensor product. The third and the fifth lists determine the dimensions along which the product will be taken. The `1` in the first list shows that we will be taking the sum along these dimensions corresponding to the third and the fifth lists. If it was a zero, we would just be returning the elementwise product along the specified dimension. What does this operation, in its current state, correspond to? Why, it is the matrix product!

Now, the more interesting part of the language is that these third and fifth lists can be as large when the input dimensionality is not this small. Note that the first list should have the same number of elements as the third and the fifth. With this method, elementwise and matrix-like products involving arbitrary numbers of dimensions could be written in a single statement. Unless you require an application in which this will work,

2.3 Just an Another Example

Let us now give some examples of more classical programs to let you become more familiar with the language. Consider the following greatest common divisor implementation to look at how control flow statements operate:

Program 3: Greatest Common Divisor

```
% Beginning of Function Declarations
function gcd (Z, X, Y);
 Z = X != Y;
 while (Z);
  if (X > Y);
   X = X - Y;
  else;
   Y = Y - X;
  end;
  Z = X != Y;
 end;
 return X;
end;
% Beginning of the Script
tensor A;
tensor B;
tensor C;
tensor Z;
A = 12;
B = 14;
A = gcd(C,A,B);
print(A);
A = 3;
B = 5;
A = gcd(C,A,B);
print(A);
```

Again, observe the way the language is structured. While loops return true as long as the last element in the relevant tensor is positive.

3 TENLAB Language Reference Manual

3.1 Introduction

This manual describes TENLAB, a programming language aimed at the use and manipulation of multidimensional arrays for scientific applications. Features defining the language include its unique handling and freeform handling of tensors, a simple syntactic structure and a powerful implementation of the general Tensor-Tensor product.

3.2 Lexical Conventions

3.2.1 Identifiers

In TENLAB, identifiers are sequences of letters, digits and the underscore character; the first character of an identifier needs to be a letter. Identifiers in TENLAB represent programmer-defined tensors. The regular expression that matches identifiers is thus

```
['a'-'z' 'A'-'Z']['a'-'z' 'A'-'Z' '0'-'9' '_']*
```

3.2.2 Keywords

Keywords are restricted to special use and as such cannot be used as identifiers. have particular use cases.

3.2.3 Functions and Blocks:

The function keyword indicates the beginning of function blocks. The end keyword marks the end of statement blocks.

```
function
end
```

3.2.4 Control Flow:

The following keywords display the control flow statements available in TENLAB:

```
if
else
while
for
```

```
return
```

3.2.5 Types:

There is a single primitive type in TENLAB, and it is `tensor`:

```
tensor
```

3.2.6 Whitespace

In TENLAB, whitespace is normally ignored, unless a comment has been detected.

3.2.7 Comments

Comments start with the % symbol and continue until the end of the `end-of-line` symbol.

3.2.8 Statement Terminator

The `;` token is used to mark the end of statements.

3.2.9 Data Types

The only type supported is called `tensor`. Tensors represent arbitrary multidimensional arrays. Elements of tensors are floating point numbers with double precision.

In addition to elements, each `tensor` also has a shape which specifies the dimensionality of the tensor. TENLAB is not strict about the limits to the number of elements as constrained through the shape parameters. That is, altering the shape of a tensor does not change anything about its content. Shape can be altered using the content of an another tensor using the built-in `reshape` function. This allows flexibility in regards to the various capabilities of TENLAB.

3.2.10 Conversions

Tensors are converted into integers internally when certain operators, specifically, equality operators would like them to be.

3.2.11 Operations

TENLAB supports the following operations:

3.2.11.1 Value Stacking

Single assignment signs indicate the stacking assignment, the stacking tensor assignment or the tensor product:

```
'='   : Assignment or Tensor Assignment
```

3.2.12 Arithmetic and Relational Operators

All arithmetic operations work as expected; in an expression, the identifier used for a tensor specifically refers to its last element. Precedence of these expressions are the same as in C.

```
'+'   : Elementwise addition
'-'   : Elementwise subtraction
'*'   : Elementwise multiplication
'/'   : Elementwise division
'=='  : Elementwise equal to
'!='  : Elementwise not equal to
'<'   : Elementwise smaller than
'<='  : Elementwise smaller than or equal to
'>'   : Elementwise greater than
'>='  : Elementwise greater than or equal to
```

3.3 Syntax

3.3.1 Program Structure

A TENLAB program consists of a number of function declarations followed by a `script` block. Both the functions and the scripts begin with a declaration of the local tensors. The `script` corresponds to the 'main' function that actually runs during the execution of a program.

3.3.1.1 Tensor Declarations

Tensors can be declared at the start of the function blocks or the script. Tensors declared within functions are not allowed to leave the scope of the function.

```
tensor identifier;
```

3.3.2 Statements

3.3.2.1 Stacking Tensor Assignment

TENLAB allows users to initialize the elements in a tensor or to stack elements to tensors using the following form:

```
identifier = tensor_list;
```

All tensors have a certain shape, and a number of elements. Shapes and the number of elements are not strictly connected.

Whereas tensors are abstracted as nested lists for users' convenience, under the hood they are one dimensional. In the list format, elements are separated using ,'s. A tensor list matches the regular expression:

```
'['(['0'-'9' ',' '[' ']' 'a'-'z' 'A'-'Z']|(((((['0'-'9'
  ]+['.']['0'-'9']*)|(['0'-'9']*['.']['0'-'9']+))(['e'
   'E']['-' '+']?['0'-'9']+)?)|(['0'-'9']+(['e' 'E']['
  -' '+']?['0'-'9']+)))+']'+
```

Checking of the validity of the sequence itself is done at compile time. Conventions for reading the tensors mostly follow the classical NumPy architecture; lists of elements in square brackets are stacked inside each other, the outermost layer corresponding to the first dimension in the shape information of the tensor.

Stacking assignments result in the concatenation of the shape information for the assigned tensors, and linear stacking of their elements to the end of the tensor in row-major ordering.

3.3.2.2 Stacking Assignment

In addition to the form above, TENLAB also allows users to initialize the elements in a tensor or to stack elements to tensors using the following form:

```
identifier = expression;
```

Which adds an element to the end of the tensor. Expressions in TENLAB support the use of literals, identifiers and paranthesized expressions connected through arithmetic and relational operators. Literals can be floating point numbers or integers, but are internally are kept as floating point numbers. Note that using an identifier within an expression only refers to its first element. After a stacking assignment, to keep track of the shape information, first dimension of the tensor is incremented by one.

3.3.2.3 Tensor Product

In addition to the aforementioned assignment methods, TENLAB also contains a powerful and terse implementation of the tensor product.

```
ID3 = {list1} {list2} ID1{list3} .* {list4} ID2 {list5}
```

Here, `ID1`, `ID2` and `ID3` refer to different tensor identifiers. `lists` are lists of integers separated with ,'s. `list1`, `list3` and `list5` should have the same length.

Elements of `list1` are either 0 or 1. `list3` and `list5` contain the corresponding indices over which the tensor product will be taken. That is, the elements in the dimension indexed by the first element of `list3` will be multiplied with the elements in the dimension indexed by the first element of `list5` and so on. `list2` and `list4` contain the dimensionalities for the tensors that are going to be used for the product.

For the indices corresponding to the 0's of `list1`, only the elementwise product is taken. If the index of `list1` is 1 for that dimension instead, the sum will be taken along the dimension after the multiplication is computed; that is, the product will act similar to a matrix product and the corresponding dimensions will collapse. Linear indexing of tensors is assumed to be column-major.

3.3.2.4 Control Flow Statements

A group of statements could be chained back to back, but only when expected by conditional statements `if`, `while` or `for`. Collectively we can refer those as `statement lists`.

Conditional statements are supported, in its most basic sense, through the `if/else` keywords. The `if` keyword could be used without an `else` like

```
if ( expression );
        statement_list;
end;
```

or it can be used in conjunction with an `else` as follows:

```
if ( expression );
        statement_list;
else;
        statement_list;
end;
```

If statements are considered true if the last element stored in the `expression` returns a positive number.

3.3.3 While Statement

The `while` keyword defines a loop statement that has the following structure:

```
while ( ID );
        statement_list;
end;
```

and the loop will continue as long as the last element of the identifier `ID` is positive. A very important feature in TENLAB is that these statements check the last value stored in the tensor.

3.3.4 For Statement

The `for` statement is in the MATLAB fashion:

```
for ID1 = ID2;
        statement_list;
end;
```

The statements in `statement_list` run through for each of the elements of `ID2`, with `ID1` getting assigned the values in `ID2` one by one.

3.3.5 Function Definitions

While different than the usual MATLAB syntax, functions are defined a similar manner to the preceding statements:

```
function function_name(identifier_list);
        statement_list;
        return ID;
end;
```

and can be called at will. The syntax for a function call is

```
function_name(identifier_list)
```

An identifier list is a list of identifiers separated by the ',' character. To promote memory safety, the returned identifier `ID` should be one of the inputs. A function could have multiple return points, using the control flow statements. Functions don't need to `return` anything.

3.3.6 Built-In Functions

3.3.6.1 MEX Input and Output

TENLAB offers two built-in functions for accessing inputs from MATLAB via the MEX interface and for the outputting formatted data.

In the C/C++ level, MEX inputs are accessed via the use of an integer corresponding to the index of the input to the MEX function. Similarly, the `input` function takes in a TENLAB tensor identifier and an integer corresponding to the equivalent input index. To be consistent with the rest of the language and following the MATLAB convention, in TENLAB these indices start from 1.

The output function works similarly. Given a tensor input and an index, at the end of the script the TENLAB program is going to output that tensor.

3.3.6.2 Printing Functions

- TENLAB programs perform printing via calls to the `print` function. The `print` function takes in a single tensor as its input argument, and prints all its values one by one. The shape information is disregarded during this.
- Similarly, the `shape` function can be used to print the contents of its sole argument, an identifier.

3.3.6.3 Clear and Clean

- The `clear` function can be used to clear the contents of a tensor. However, it does not remove a tensor completely from memory.
- The more powerful `clean` function can be used to clean the tensor along with its identifier from the memory completely. Both of these functions take a single tensor as input.

3.3.6.4 Pop, Dequeue, Length and Set

- The `pop` function can be used to remove the last element a tensor. The `dequeue` function can be used to remove the first element. `length` function can be used to add the current number of elements to the tensor. All of these functions take a single tensor identifier as input.
- The `set` function takes three tensors, and can be used to change the element of the first tensor using the shape of the first tensor as the shape and the content of the second tensor as the indices. The last element of the third tensor is the element that gets set. If the indices correspond to a larger index than the ones in the tensor, enough 0's are appended to the content of the tensor to broadcast the shape information.

4 Project Plan

4.1 Planning Process

As a group, we have decided to seperate the work into different independent parts that could then be compiled into a whole without any need of rigorous. We had regular weekly meetings with Professor Edwards and we discussed our progress as a team regularly as well.

4.2 Specification

From the beginning of the project, we had a clear niche audience to target for the project along with a huge number of backup plans for possible bugs and design issues from which our future architectures could perhaps not recover. Thanks to the relative simplicity and ambiguity of the initial proposal for our language, our meetings with Professor Edwards helped us organically consider alternatives and reshape the the various subtle details of the language and how some parts of the code could be rendered more optimized as we did not think that we would be able to match the performance of languages like MATLAB without the use of extrinsic libraries or clever tricks, a very substantial threat which would render our language redundant.

Due to several problems we had with memory handling in C in regards to dynamic array structures that could conceivably be expected to contain and manipulate large blocks of data with relative ease for the machine learning applications as well as the comparative inexperience of our team members in such matters, we have ultimately decided to find a compromise between simplicity and extensibility that renders TENLAB rather unique compared to its alternatives.

4.3 Development Process

Due to the relative simplicity of our language in regards to the scanner and parser, those modules were created very early on during the writing of the language reference manual in order to promote efficiency in the future. Code generation for functions as well as the control flow statements were similarly completed long before the finalization of the rest of the subtler specifics of the language.

4.4 Team Responsibilities

At the beginning of the project, Yusuf Cem Subakan wanted the leadership position that would entail the responsibilities of *Language Guru* and *Project Manager*. Even though he was against the idea of weekly meetings, suggesting that they do not work and stating that he always finished the group projects he took part in himself, thanks to the directives in the course announcements he agreed to weekly meetings with Professor Edwards. Due to the various problems he had over the course of the semester, the roles of the team members needed to be more and more fluid over time. Other members of the team held weekly meetings in addition to those with Professor Edwards to discuss the situation of the codebase and how it could be improved.

Team Member	**Responsibility**
Mehmet Turkcan	Code Generation, Test Case Generation, C Libraries, Testing Automation, Specification, Documentation
Yusuf Cem Subakan	Language Guru, Project Manager
Dallas Randal Jones	Testing Automation, Specification, Documentation

4.5 Development Environment

The full list of the software we have employed through the development of our language are as follows:

- **Bitbucket Git Repository:** We have set up our git environment the day we formed our group, and have used it constantly.
- **OCaml Version 4.02.3:** For the main TENLAB compiler that outputs `.c` code.
- **GCC:** For building the `.c` output.
- **Cygwin64:** For the tests on Windows.
- **MATLAB Version R2015a:** For the testing of the `mex` interface.
- **Microsoft Visual C++ 2013 Professional:** For building and testing the `mex` interface.
- **Latex:** We like nice-looking reports.
- **Gimp:** For the design and creation of the project logo.

Development took place on Windows 10 and Ubuntu, and across a large range of hardware configurations from AWS g2.8xlarge instances to Surface 3.

4.6 Programming Style Guide

We have generally adhered on the following guidelines during the coding of the language:

- We have followed the OCaml formatting style, though portions of the code have some differences that rendered it easier for us to reason about certain subsets of the code during the coding stage. We have chosen to use two spaces for indentation.
- We have fully adhered to the 80-column rule throughout the code.
- All parts of the program and the internal functions were named according to their purpose so that they would be easy to read and understand for a newcomer.

- Underscores were utilized for the naming of the variables.
- For the C backend, we have decided to use descriptive and long names for the extendability of the language in the future. As the majority of the length came from the strings and due to time constraints, we have not adhered to the 80-column rule throughout those files.

4.7 Project Timeline

Date	**Milestone**
February 3rd	Bitbucket Repository Created
February 23rd	First Draft of the Language Reference Manual
March 1st	First draft of the scanner and the parser
April 4th	Code generation successful
April 6th	Various Hello World programs working
May 5th	First version of the C backend
May 9th	Mex interface complete, first draft of the Final Report
May 11th	Project presentation and submission of the Final Report

Project Log is provided after the appendix.

5 Architectural Design

5.1 Compiler and Block Diagram:

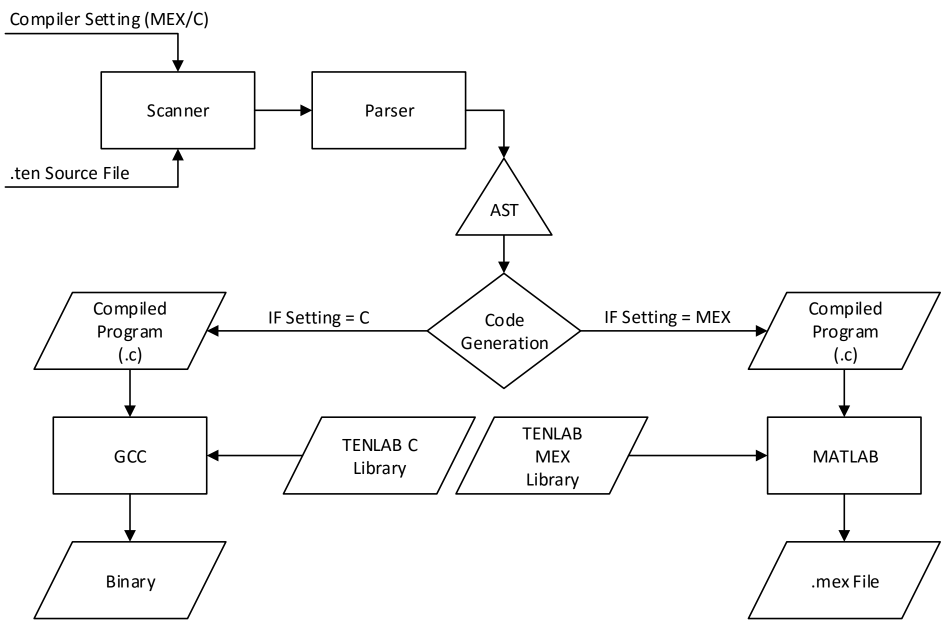

Figure 1: Architecture for the compilation of a TENLAB program, showing the two different ways in which programs can be built.

The architecture of the TENLAB compiler consists of a scanner, a parser, an AST module (termed Ccode for convenience), a code generator and a C header to glue everything together. The scanner and the parser form the front end of the TENLAB compiler and the code generator forms the back end along with the C back end. All of these components except the C headers are written in OCaml.

Entry point to the compiler is the `tenlab.ml` file, which sequentially calls the components of the compiler sequentially. Firstly, the input is passed to `scanner.ml` which generates tokens. Those are then passed to `parser.ml` and an Abstract Syntax Tree is generated using the datatypes defined in `ccode.ml`. The Abstract Syntax Tree is then passed to the `compile.ml`, which performs most of the C code generation, matching some of the more common structures with an intermediate format. This format is then processed into a string using a list of definitions kept in ccode.ml and the compiled C code is outputted. This code can then be compiled into binary using. Due to the fundamental differences between TENLAB and C, the C code output also requires a custom-built library written in C, called `tenlab_preamble.c`, to work.

Finally, to be fully compatible with MATLAB, a version of the library, called `tenlab_preamble.cpp`, is included which has very minor differences. For files compiled using the MEX compilation option, at this time this library is required.

5.2 Scanner:

- Relevant Files: `scanner.mll`

Written in ocamllex, the scanner takes a TENLAB file as input and tokenizes that input into literals, identifiers and keywords. Tokens created by the scanner are passed to the parser.

5.3 Parser:

- Relevant Files: `parser.mly`, `ccode.ml`

Written in ocamlyacc, the parser `parser.mly` gets a series of tokens from the scanner and then generates an Abstract Syntax Tree (AST) using the grammar declared in the `ccode.ml` file.

5.4 Code Generation:

- Relevant Files: `compile.ml`, `ccode.ml`

TENLAB compilation continues with a single-pass, depth-first traversal of the Abstract Syntax Tree generated. Parts of the code for which the corresponding `.c` code could be generated immediately are converted; to abbreviate the generated code to expedite programming, some of the code is converted into an intermediate format. Finally, a second pass is made over the now-linearized list using a list of definitions that are kept in the ccode.ml file for convenience to turn the compile.ml output into a string and thus generate the compilable `.c` code.

Due to a number of fundamental differences between MEX and regular C, the compiler also needs to know the type of output that's going to be generated. Headers for the two targets as well as the entry functions are different and are decided upon by looking at the directives given to the compiler.

5.5 C Functions:

- Relevant Files: `tenlab_preamble.c`, `tenlab_preamble_mex.cpp`

These collections of functions include a number of low level functions that allow TENLAB to avoid various memory problems that could come up. Generated `.c` code automatically includes the relevant file in the header.

Every component was built and integrated by Mehmet; the initial version of the tensor product code generation submodule in the was built by Cem and was later integrated into the language by Mehmet.

6 Testing

6.1 From Source to Target

Let us begin with a two-dimensional matrix product:

TENLAB Test File 1: Matrix Product

```
tensor X;
tensor Y;
tensor Z;
X = [[3,4],[4,5],[6,7]];
Y = [[1,2,3],[4,5,6]];
Z = [[0,0,0],[0,0,0],[0,0,0]];
Z = {1} {3,2} X {2} .* {2,3} Y {1};
print(Z);
```

and let us give the output:

TENLAB Test File 2: Matrix Product

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "tenlab_preamble.c"
// Stats: 3 Script Variables;

int main(){
TENLAB_Tensor Z;
TENLAB_Tensor_create(&Z);
TENLAB_Tensor Y;
TENLAB_Tensor_create(&Y);
TENLAB_Tensor X;
TENLAB_Tensor_create(&X);
TENLAB_assign(&X,3);
TENLAB_assign(&X,4);
TENLAB_assign(&X,4);
TENLAB_assign(&X,5);
TENLAB_assign(&X,6);
TENLAB_assign(&X,7);
TENLAB_add_shape(&X,2);
TENLAB_add_shape(&X,3);
;
TENLAB_assign(&Y,1);
TENLAB_assign(&Y,2);
TENLAB_assign(&Y,3);
TENLAB_assign(&Y,4);
TENLAB_assign(&Y,5);
TENLAB_assign(&Y,6);
TENLAB_add_shape(&Y,3);
TENLAB_add_shape(&Y,2);
;
TENLAB_assign(&Z,0);
TENLAB_assign(&Z,0);
TENLAB_assign(&Z,0);
TENLAB_assign(&Z,0);
```

```
TENLAB_assign(&Z,0);
TENLAB_assign(&Z,0);
TENLAB_assign(&Z,0);
TENLAB_assign(&Z,0);
TENLAB_assign(&Z,0);
TENLAB_add_shape(&Z,3);
TENLAB_add_shape(&Z,3);
;
for(int TENLAB_i1=0;TENLAB_i1<3;TENLAB_i1++) {
 for(int TENLAB_j2=0;TENLAB_j2<3;TENLAB_j2++) {
 for(int TENLAB_k1=0;TENLAB_k1<2;TENLAB_k1++) {
 Z.Content[TENLAB_i1+TENLAB_j2*3] = Z.Content[TENLAB_i1+
     TENLAB_j2*3] + X.Content[TENLAB_i1+TENLAB_k1*3] * Y.Content
     [TENLAB_k1+TENLAB_j2*2];
}
 }
 }
 TENLAB_Tensor_print(Z);
}
```

Let continue with a simpler GCD Algorithm:

TENLAB Test File 3: GCD

```
% Beginning of Function Declarations
function gcd (Z, X, Y);
 Z = X != Y;
 while (Z);
  if (X > Y);
   X = X - Y;
  else;
   Y = Y - X;
  end;
  Z = X != Y;
 end;
 return X;
end;
% Beginning of the Script
tensor A;
tensor B;
tensor C;
tensor Z;
A = 12;
B = 14;
A = gcd(C,A,B);
print(A);
A = 3;
B = 5;
A = gcd(C,A,B);
print(A);
```

and the target output:

TENLAB Test File 4: GCD

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include "tenlab_preamble.c"
// Stats: 4 Script Variables;

TENLAB_Tensor gcd(TENLAB_Tensor Z,TENLAB_Tensor X,TENLAB_Tensor
    Y){
TENLAB_assign(&Z,X.Content[X.cur_content_length-1]!=Y.Content[Y.
    cur_content_length-1]);
while (Z.Content[Z.cur_content_length-1]>0) {
if (X.Content[X.cur_content_length-1]>Y.Content[Y.
    cur_content_length-1]){
TENLAB_assign(&X,X.Content[X.cur_content_length-1]-Y.Content[Y.
    cur_content_length-1]);

}
else
{
TENLAB_assign(&Y,Y.Content[Y.cur_content_length-1]-X.Content[X.
    cur_content_length-1]);
}
TENLAB_assign(&Z,X.Content[X.cur_content_length-1]!=Y.Content[Y.
    cur_content_length-1]);
}
return X;
}
int main(){
TENLAB_Tensor Z;
TENLAB_Tensor_create(&Z);
TENLAB_Tensor C;
TENLAB_Tensor_create(&C);
TENLAB_Tensor B;
TENLAB_Tensor_create(&B);
TENLAB_Tensor A;
TENLAB_Tensor_create(&A);
TENLAB_assign(&A,12);
TENLAB_assign(&B,14);
TENLAB_assign(&A,gcd(C,A,B));
TENLAB_Tensor_print(A);
TENLAB_assign(&A,3);
TENLAB_assign(&B,5);
TENLAB_assign(&A,gcd(C,A,B));
TENLAB_Tensor_print(A);
}
```

6.2 Test Suites and Automation

The testing of TENLAB programs was automated using a modified version of the `MICROC` compiler's bash test script provided by Professor Edwards. This modified testing suite, now termed `testlab`, was among the first pieces of code written for the compiler. Throughout the development of the language, we have first generated a number of test cases that the language should compile and what the language, and then worked on towards making those programs work.

At first, the testing suite only allowed the checking of pass cases; modifications to allow the testing of the fail cases were done shortly afterwards during the development. As the language began to mature and depend heavily on the usability of the custom C library that allows the language to operate, a separate test suite with the same capabilities was eventually developed as well, now termed `testc`.

The current statistics for the testing files is as follows: There are 18 test cases for `testc` and 35 for `testlab`. The source package also includes 4 low level unit tests to show how TENLAB handles of the built-in `input` and `output` functions.

In addition to these automated testing solutions, there were some basic unit testing suites for the tensor parsing and tensor product part of the language that were written during development. Tensor matching was successfully integrated into the language in a straightforward manner. The tensor product suite that was in our initial specification depended on information that would not be available to the compiler unless the language was severely constrained; even though the relevant team member responsible was uninterested in working on the project, the model was integrated to meet the specifications with some sacrifices and the relevant team member was valuable in this endeavour. As those tests are no longer needed, they are not provided.

Handling the garbage collection issues became a colossal priority around the end of the project, which required a significant portion of the language to be changed and required the introduction of Struct's to keep track of everything internally; before that, the language had relied on the use of two dynamic arrays per tensor to keep track of everything at a lower level. It was thanks to our automation system that we were able to address some potentially catastrophic memory issues.

Source files for the testing scripts are included in the Appendix. All of the testing was done by Mehmet.

7 Lessons Learned & Advice

7.1 Mehmet Turkcan:

7.1.1 Lessons Learned:

Whereas I had worked in projects in which the addition of a small feature could break significant sections of the program before, I had never taken part in a project like this in which every single section of the code should be kept in mind during the programming process. Building an automated test suite very early on and writing up a list of test cases beforehand was definitely the best choice we made in regards to the handling of the project. Certainly, the course has changed my mind on the importance of testing.

Lastly, I must say that OCaml is actually really fun to work with; however, it did take me some time to truly understand the language and its capabilities.

7.1.2 Advice:

It is important to make sure that the team members are comfortable with the tools and the programming languages that are going to be used during the project and to determine the strengths and the weaknesses of the team members well before the submission of the initial proposal.

Secondly, versatility of the team members is of immense importance. Specifically, it is crucial that all group members are able to learn and get comfortable with OCaml as well as the various other languages and tools that are to be used during the development very early on. Every member of the team should be aware of the details of the implementation and the restrictions imposed by the architecture at all times and be capable of handling their responsibilities, especially as the term continues and people begin to encounter subtle roadblocks, which have the potential to bottleneck the progress of the whole project.

My final advice to future groups is to be very selective during the teaming up process and to prefer people you already know and have worked with together in the past: people change with time, can have stressful periods in their lives and even those with otherwise impressive achievements may lack the versatility the project demands.

8 Appendix

TENLAB Source File 1: scanner.mll

```
{ open Parser }

rule token = parse
  [' ' '\t' '\r' '\n']  { token lexbuf }            (* Whitespace *)
| "%"  { comment lexbuf }           (* Comments   *)
| '('  { LPAREN }
| ')'  { RPAREN }
| '['  { LNPAREN }
| ']'  { RNPAREN }
| '{'  { LBRACE }
| '}'  { RBRACE }
| ':'  { COLON }
| ".*"  { TENPROD }
| ';'  { SEMI }
| ','  { COMMA }
| '+'  { PLUS }
| '-'  { MINUS }
| '*'  { TIMES }
| '/'  { DIVIDE }
| '='  { ASSIGN }
| "=="  { EQ }
| "!="  { NEQ }
| '<'  { LT }
| "<="  { LEQ }
| ">"  { GT }
| ">="  { GEQ }
| "if"  { IF }
| "function"  { FUNCTION }
| "end"  { END }
| "else"  { ELSE }
| "for"  { FOR }
| "while"  { WHILE }
| "return"  { RETURN }
| "tensor"  { TENSORDEF }
| "clear"  { CLEAR }
| "clean"  { CLEAN }
| '['(['0'-'9' ',' '[' ']' 'a'-'z' 'A'-'Z']|
  ((((['0'-'9']+['.']['0'-'9']*)|(['0'-'9']*['.']['0'-'9']+))
  (['e' 'E']['-' '+']?['0'-'9']+)?)|(['0'-'9']+(['e' 'E']
  ['-' '+']?['0'-'9']+))))+']'+ as lxm  { TENSOR(lxm) }
| ['0'-'9']+ as lxm  { LITERAL(lxm) }
| ((((['0'-'9']+['.']['0'-'9']*)|(['0'-'9']*['.']['0'-'9']+))
  (['e' 'E']['-' '+']?['0'-'9']+)?)|(['0'-'9']+(['e' 'E']
  ['-' '+']?['0'-'9']+))) as lxm  { LITERAL(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
  ['\r' '\n']  { token lexbuf }
| _  { comment lexbuf }
```

TENLAB Source File 2: parser.mly

```


%{ open Ccode %}

%token SEMI LPAREN RPAREN LNPAREN RNPAREN LBRACE RBRACE COMMA COLON TENPROD
%token PLUS MINUS TIMES DIVIDE ASSIGN
%token EQ NEQ LT LEQ GT GEQ
%token RETURN IF ELSE FOR WHILE TENSORDEF CLEAR CLEAN FUNCTION END
%token <string> LITERAL
%token <string> ID
%token <string> TENSOR
%token EOF

%nonassoc NOELSE
%nonassoc ELSE
%right ASSIGN
%left EQ NEQ
%left LT GT LEQ GEQ
%left PLUS MINUS
%left TIMES DIVIDE

%start program
%type <Ccode.program> program

%%

program:
  decls EOF { $1}
  | decls main_decl { fst $1, ($2 :: snd $1) }
decls:
   /* nothing */ { [], [] }
 | decls vdecl { ($2 :: fst $1), snd $1 }
 | decls fdecl { fst $1, ($2 :: snd $1) }


 main_decl:
   stmt_list
     { { fname = "main";
   formals = [];
   locals = [];
   body = List.rev $1 } }

fdecl:
   FUNCTION ID LPAREN formals_opt RPAREN SEMI vdecl_list stmt_list END SEMI
     { { fname = $2;
   formals = $4;
   locals = List.rev $7;
   body = List.rev $8 } }

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:
   TENSORDEF ID SEMI { $2 }

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

stmt:
    expr SEMI  { Expr($1) }
  | RETURN expr SEMI
      { Return($2) }
  | CLEAR ID SEMI
      { Clear($2) }
  | CLEAN ID SEMI
      { Clean($2) }
  | ID ASSIGN LBRACE num_list RBRACE LBRACE num_list RBRACE ID LBRACE num_list RBRACE
      TENPROD LBRACE num_list RBRACE ID LBRACE num_list RBRACE SEMI
      { BuildTensorProd( $1, List.rev $4, List.rev $7, $9,
      List.rev $11, List.rev $15, $17, List.rev $19 ) }
  | IF LPAREN expr RPAREN SEMI stmt_list END SEMI %prec NOELSE
      { If($3, Block(List.rev $6), Block([])) }
  | IF LPAREN expr RPAREN SEMI stmt_list ELSE SEMI stmt_list END SEMI
      { If($3, Block(List.rev $6), Block(List.rev $9)) }
  | FOR ID ASSIGN ID SEMI stmt_list END SEMI
      { For($2, $4, Block(List.rev $6)) }
  | WHILE LPAREN expr RPAREN SEMI stmt_list END SEMI
      { While($3, Block(List.rev $6)) }

expr:
    LITERAL                       { Literal($1) }
  | ID                            { Id($1) }
  | TENSOR                        { TensorGet($1) }
  | expr PLUS   expr              { Binop($1, Add,   $3) }
  | expr MINUS  expr              { Binop($1, Sub,   $3) }
  | expr TIMES  expr              { Binop($1, Mult,  $3) }
  | expr DIVIDE expr              { Binop($1, Div,   $3) }
  | expr 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) }
  | ID ASSIGN   expr              { Assign($1, $3) }
  | ID LPAREN actuals_opt RPAREN { Call($1, $3) }
  | LPAREN expr RPAREN            { $2 }

actuals_opt:
    /* nothing */                 { [] }
  | actuals_list                  { List.rev $1 }
```

```

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

num_list:
    LITERAL                              { [int_of_string $1] }
  | num_list COMMA LITERAL          { (int_of_string $3) :: $1 }
```

TENLAB Source File 3: ccode.ml

```
(* TENLAB AST Module by Mehmet Kerem Turkcan *)
(* Based on the corresponding MICROC Module  *)

type op = Add | Sub | Mult | Div | Equal | Neq | Less | Leq | Greater | Geq

type expr =
    Literal of string
  | Id of string
  | TensorGet of string
  | Binop of expr * op * expr
  | Assign of string * expr
  | Call of string * expr list
  | GetTensorElement of string * string
  | Noexpr

type stmt =
    Block of stmt list
  | Expr of expr
  | BuildTensorProd of string * int list * int list * string * int list
      * int list * string * int list
  | CreateTensor of string * string
  | Return of expr
  | Clear of string
  | Clean of string
  | If of expr * stmt * stmt
  | For of string * string * stmt
  | While of expr * stmt

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

type program = string list * func_decl list

let rec string_of_expr = function
    Literal(l) -> l
  | Id(s) -> s ^ ".Content[0]"
  | TensorGet(s) -> s
  | Binop(e1, o, e2) ->
      string_of_expr e1 ^ " " ^
      (match o with
  Add -> "+" | Sub -> "-" | Mult -> "*" | Div -> "/"
      | Equal -> "==" | Neq -> "!="
      | Less -> "<" | Leq -> "<=" | Greater -> ">" | Geq -> ">=" ) ^ " " ^
      string_of_expr e2
  | Assign(v, e) -> v ^ " = " ^ string_of_expr e
  | Call(f, el) ->
      f ^ "(" ^ String.concat ", " (List.map string_of_expr el) ^ ")"
  | Noexpr -> ""
  | _ -> raise (Failure ("TENLAB Error: Something impossible was
```

```
observed."))

type bstmt =
    (* Direct Output of Numeric Literal *)
    Lit of string
  (* Direct Output of String *)
  | DirectOut of string
  (* Tensor of String *)
  | TensorGet of string
  (* Tensor Variable of String *)
  | TempVar of string
  (* Tensor Declaration of String *)
  | VarDeclare of string
  (* Tensor Operation of Strings *)
  | TensorOpt of string * string * string
    (* Direct Output of Tensor *)
  | TensorAssign of string
  (* Free Memory of Temporary Variable *)
  | TempMemFree of int
  (* Get Size of Temporary Variable *)
  | TempGetSize of int
  (* Direct Output of String *)
  | TensorSub of string
    (* Call Function by Explicit Name *)
  | FunctionCall of string
    (* Return Values at the End of Function *)
  | ReturnValue of int
    (* Clear a Tensor *)
  | ClearValue of string
  (* Clean a Tensor from memory *)
  | CleanValue of string
    (* Return Values at the End of Function *)
  | VarAssign of int
    (* Put Right Paranthesis *)
  | Parend
    (* End of Function *)
  | Curlend
    (* Begin If Statement *)
  | Beginf
    (* Put a Comma *)
  | Comma
    (* End Line and Begin New Line *)
  | Lineend
    (* Begin Curl for Blocks and Start a New Line*)
  | Curlbegin
    (* Arithmetic Operations *)
  | Bin of op

type prog = {
    num_globals : int;                          (* Number of global variables *)
    text : bstmt array;                         (* Code for all the functions *)
  }

(*
Match Implicit Blocks
*)

```

```
let string_of_stmt = function
    Lit(i) -> i
  | DirectOut(i) -> i
  | TempVar(i) ->
      "TENLAB_Tensor temp" ^ i ^ ";\nTENLAB_Tensor_create(&" ^ i ^ ");\n"
  | TensorGet(i) -> i
  | TempMemFree(i) -> "TENLAB_Tensor_destroy(&temp" ^
    string_of_int i ^ ")"
  | TempGetSize(i) -> "temp" ^ string_of_int i ^ ".cur_content_length"
  | TensorSub(i) -> i
  | VarDeclare(i) -> i
  | TensorAssign(i)  -> i ^ " = "
  | TensorOpt(i,j,k)  -> i ^ j ^ k
  | FunctionCall("input")  -> "TENLAB_Tensor_populate_from_MEX(prhs,&"
  | FunctionCall("output")  -> "TENLAB_Tensor_to_MEX(plhs,&"
  | FunctionCall("pop")  -> "TENLAB_pop_element(&"
  | FunctionCall("dequeue")  -> "TENLAB_dequeue_element(&"
  | FunctionCall("reshape")  -> "TENLAB_Tensor_reshape(&"
  | FunctionCall("print")  -> "TENLAB_Tensor_print("
  | FunctionCall("shape")  -> "TENLAB_Tensor_shape_print("
  | FunctionCall("set")  -> "TENLAB_add_element_at_specific_position(&"
  | FunctionCall("length")  -> "TENLAB_add_length(&"
  | FunctionCall("main")  -> "int main("
  | FunctionCall(i)  -> i ^ "("
  | VarAssign(i)  -> "double var" ^ string_of_int i ^ " = "
  | ReturnValue(i)  -> "return "
  | ClearValue(i) ->
      "TENLAB_Tensor_destroy(&" ^ i ^ ");\nTENLAB_Tensor_create(&" ^ i ^ ")"
  | CleanValue(i)  -> "TENLAB_Tensor_destroy(&" ^ i ^ ");"
  | Bin(Add)  -> "+"
  | Bin(Sub)  -> "-"
  | Bin(Mult)  -> "*"
  | Bin(Div)  -> "/"
  | Bin(Equal)  -> "=="
  | Bin(Neq)  -> "!="
  | Bin(Less)  -> "<"
  | Bin(Leq)  -> "<="
  | Bin(Greater)  -> ">"
  | Bin(Geq)  -> ">="
  | Parend  -> ")"
  | Curlend  -> "}\n"
  | Beginf  -> "if ("
  | Comma  -> ","
  | Lineend  -> ";\n"
  | Curlbegin  -> "{\n"

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

(*
Generate Program Text
*)

(*
TENLAB Preamble:
```

```
Populate for the Language Dependencies via C Libraries and the TENLAB C Backend
*)
let string_of_prog p =
  "#include <stdio.h>\n#include <math.h>\n#include <stdlib.h>\n" ^
  "#include \"tenlab_preamble.c\"\n// Stats: " ^ string_of_int p.num_globals ^
  " Script Variables;\n\n" ^
  let funca = Array.mapi
      (fun i s -> "" ^ string_of_stmt s) p.text
  in String.concat "" (Array.to_list funca)

(*
Generate Program Text for MEX-Supported Programs
*)

(*
TENLAB Preamble:
Populate for the Language Dependencies via C Libraries and the TENLAB C Backend
*)
let string_of_prog_mex p =
"#include <stdio.h>\n#include <math.h>\n#include <stdlib.h>\n#include <mex.h>\n"
^ "#include \"tenlab_preamble_mex.cpp\"\n// Stats: " ^
string_of_int p.num_globals ^
  " Script Variables;\n\n" ^
  let funca = Array.mapi
      (fun i s -> "" ^ string_of_stmt s) p.text
  in String.concat "" (Array.to_list funca)
```

TENLAB Source File 4: compile.ml

```ocaml
(* TENLAB COMPILATION Module by Mehmet Kerem Turkcan and Yusuf Cem Subakan *)
(* Based on the corresponding MICROC Module                                   *)

open Ccode
open Str

module StringMap = Map.Make(String)
(*
 This converts the dimension matching list into a binary list
*)
let rec list2bin lst dims cnt =
  match lst,dims with
    | [],[] -> []
    | _,[] -> []
    | [],hd::tl2 -> 0::(list2bin [] tl2 (cnt+1))
    | hd::tl,hd2::tl2 -> if (cnt=hd) then 1::(list2bin tl tl2 (cnt+1))
          else 0::(list2bin lst tl2 (cnt+1));;
(*
This function fills the unmatched indices of a tensor in a list
*)
let rec binary2inds match_dims len charac =
  match match_dims with
    | [] -> []
    | hd::tl -> if (hd=0) then (charac^(string_of_int ( len -
          (List.length tl))))::(binary2inds tl len charac)
          else binary2inds tl len charac;;
(*
This function extracts the dimension limits for specified limits in the binary
list binsA / one_or_zero input specifies whether matched/non-matched indices
should be picked
*)
let rec binary2lims binsA dims one_or_zero =
  match binsA,dims with
    | [],[] -> []
    | _,[] -> []
    | [],_ -> []
    | hd::tl,hd2::tl2 -> if (hd=one_or_zero) then
          hd2::(binary2lims tl tl2 one_or_zero)
          else binary2lims tl tl2 one_or_zero;;
(*
This functions writes the for loops, for the observable indices of a tensor in a list
*)
let rec writefors inds dims =
  match inds,dims with
    | [],[] -> []
    | hd::tl,[] -> []
    | [],hd2::tl2 -> []
    | hd::tl,hd2::tl2 -> ("for(int "^hd^"=0;"^hd^"<"^(string_of_int hd2)^
          ";"^hd^"++) { \n " )::(writefors tl tl2);;
(*
```

```
This produces a list of length len consisting of all n's
*)
let rec all_n_list n len cnt =
  if (cnt<=len) then n::(all_n_list n len (cnt+1))
  else [];;
(*
This function is needed for linear indexing
*)
let rec multiply_lims lims ind cnt =
  match lims with
    | [] -> ""
    | hd::tl -> if (cnt<ind-1) then (string_of_int hd) ^ "*" ^
        (multiply_lims tl ind (cnt+1))
        else if (cnt = ind-1) then string_of_int hd
        else "";;
(*
This function writes the linear indices given the index and dimensions list
*)
let rec linear_indices inds lims len cnt=
  match inds with
    | [] -> ""
    | hd::tl -> if (cnt = 1 && len > 1) then hd ^ "+" ^
        (linear_indices tl lims len (cnt+1))
        else if (cnt = 1 && len = 1) then hd
        else if (cnt > 1 && cnt<len && len>1) then
        hd^"*"^(multiply_lims lims cnt 1) ^ "+" ^
        (linear_indices tl lims len (cnt+1))
        else hd^"*"^(multiply_lims lims cnt 1) ^
        (linear_indices tl lims len (cnt+1));;
(*
This function negates a binary list
*)
let rec negate_list lst =
  match lst with
  | [] -> []
  | hd::tl -> (1-hd)::(negate_list tl);;


(*
This function forms the indices/limits given the binary matching list binsA,
observable indices obsinds and matched indices matchedinds
*)
let rec form_indsA binsA obsinds matchedinds cnt1 cnt2 =
  match binsA with
    | [] -> []
    | hd::tl -> if (hd=0) then (List.nth obsinds cnt1)::
      (form_indsA tl obsinds matchedinds (cnt1+1) cnt2)
      else (List.nth matchedinds cnt2)::
      (form_indsA tl obsinds matchedinds cnt1 (cnt2+1));;
(*
This function finds the index of the element x in list lst -
the index starts from 1
*)
let find_index lst x =
  let rec find_x_index lst x cnt =
    match lst with
      | [] -> -1
```

```
      | hd::tl -> if (hd=x) then cnt
        else find_x_index tl x (cnt+1)
        in find_x_index lst x 0;;
(*
This function forms the indices of B
*)
let rec form_indsB binsB obsinds matchedinds matchB =
  let rec form binsB obsinds matchedinds matchB cnt1 cnt2 =
    match binsB with
    | [] -> []
    | hd::tl -> if (hd=0) then
    (List.nth obsinds cnt1)::
    (form tl obsinds matchedinds matchB (cnt1+1) (cnt2+1))
      else (List.nth matchedinds (find_index matchB cnt2))::
      (form tl obsinds matchedinds matchB cnt1 (cnt2+1))
        in form binsB obsinds matchedinds matchB 0 1;;

let rec concat_strings_inlist lst =
        match lst with
        | [] -> ""
        | hd::tl -> hd^(concat_strings_inlist tl);;

(*
Write the 'observable' for loops for tensor A in a list called forsA
*)
let create_tensor_product_code c sum_or_diag dimsA a matchA dimsB b matchB =
  let binsA = list2bin matchA dimsA 1 in
  let indsA = binary2inds binsA (List.length binsA) "TENLAB_i" in
  let limsA = binary2lims binsA dimsA 0 in
  let forsA = writefors indsA limsA in

  (*
  Write the 'observable' for loops for tensor B ins a list called forsB
  *)
  let binsB = list2bin (List.sort compare matchB) dimsB 1 in
  let indsB = binary2inds binsB (List.length binsB) "TENLAB_j" in
  let limsB = binary2lims binsB dimsB 0 in
  let forsB = writefors indsB limsB in

  (*
  Write the 'matched' for loops
  *)
  let inds_matched_all = binary2inds
    (*
    Get inds
    *)
    (all_n_list 0 (List.length sum_or_diag) 1 ) (List.length sum_or_diag)
    "TENLAB_k" in
  let inds_matched_observable =
    binary2inds sum_or_diag (List.length sum_or_diag) "TENLAB_k" in
  let inds_matched_collapsed =
    binary2inds (negate_list sum_or_diag) (List.length
```

```
    sum_or_diag) "TENLAB_k" in
(*
Get the limits
*)
let lims_matched_all = binary2lims binsA dimsA 1 in
let lims_matched_observable = binary2lims sum_or_diag lims_matched_all 0 in
let lims_matched_collapsed = binary2lims sum_or_diag lims_matched_all 1 in
(*
Write for's
*)
let fors_matched_observable =
  writefors inds_matched_observable lims_matched_observable in
let fors_matched_collapsed =
  writefors inds_matched_collapsed lims_matched_collapsed in
(*
Concatanete to get all fors
*)
let all_fors = forsA @ forsB @ fors_matched_observable @
  fors_matched_collapsed in
(*
get the indices and limit of the term C by concatenating
observable indices/limits
*)
let all_inds_C = indsA @ ( indsB @ inds_matched_observable) in
let all_lims_C = limsA @ ( limsB @ lims_matched_observable) in
(*
write down the linear indices for C
*)
let term_C = linear_indices all_inds_C all_lims_C
  (List.length all_inds_C) 1 in
(*
Get all indices for A
*)
let all_inds_A = form_indsA binsA indsA inds_matched_all 0 0 in
(*
Write down what we have for A in linear indices
*)
let term_A = linear_indices all_inds_A dimsA (List.length all_inds_A) 1 in

let all_inds_B = form_indsB binsB indsB inds_matched_all matchB in
(*
Write down what we have for B in linear indices
*)
let term_B = linear_indices all_inds_B dimsB (List.length all_inds_B) 1 in
(*
Aggregate the terms inside fors
*)
let theterm_inside_fors = c^".Content["^term_C^"] = " ^
  c ^ ".Content[" ^ term_C ^ "] + " ^ a ^ ".Content[" ^ term_A ^ "] * " ^
  b ^ ".Content[" ^ term_B ^ "]; \n" in
(*
Form the closing brackets
```

```
  *)
  let closing_brackets = all_n_list "} \n " ((List.length all_fors)-1) 0 in
  (*
  Put everything inside a big string - the output dimensions are in all_lims_C
  *)
  let everything =
    (concat_strings_inlist all_fors) ^ theterm_inside_fors ^
    (concat_strings_inlist closing_brackets) in everything;;



let str_crop_last_char x_in =
  if x_in = "" then "" else
  String.sub x_in 0 ((String.length x_in) - 1)

let get_string_length x_in = string_of_int((( String.length x_in)-1)/2);;

let get_tensor_dimension e =
  let temp = List.filter (fun x -> (String.length x)>0) (List.map (function
    | Str.Delim s -> s
    | _ -> "") e) in
  let mapper x_in=  ((String.length x_in)-1)/2 in
  let temp2 = List.sort_uniq (fun x y -> if x > y then 1 else 0)
    (List.map (mapper) temp) in
  temp2;;

let get_tensor_blocks x_in =
  Str.full_split (Str.regexp "\\[\\([0-9]+[',']\\)+[0-9]+\\]") x_in;;

let rec build_tensor e =
  if (List.length e) > 1 then
  let temp = (List.map (function
                   | Str.Delim s -> (get_string_length s)
                   | Str.Text  s ->  s) e) in
  let temp1 = build_tensor (get_tensor_blocks (String.concat "" temp)) in
  let temp2 = get_tensor_dimension e in
  temp2::temp1 else
  [];;

let get_tensor_elements x_in =
  List.map int_of_string (Str.split (Str.regexp "[^0-9]+") x_in);;


let declare_tensor_in_C name_in content_in=
  let temp1 = (List.concat (build_tensor
    (get_tensor_blocks ("[" ^ content_in ^ "]")))) in
  let temp3 = List.fold_left (fun x1 x2 -> x1 ^ x2 ^ ");\nTENLAB_assign(&" ^
   name_in ^ ",") ("TENLAB_assign(&" ^ name_in ^ ",")
    ( (Str.split (Str.regexp "[^0-9]+") content_in)) in
  let temp4 = String.sub temp3 0
    ((String.length temp3)-17-(String.length name_in)) in
```

```
  let temp5 = List.fold_left (fun x1 x2 -> x1 ^ (string_of_int x2) ^
    ");\nTENLAB_add_shape(&" ^ name_in ^ ",")
    ("\nTENLAB_add_shape(&" ^ name_in ^ ",") temp1 in
  let temp6 = String.sub temp5 0
    ((String.length temp5)-19-(String.length name_in)) in
  "TENLAB_Tensor " ^ name_in ^
  ";\nTENLAB_Tensor_create(&" ^ name_in ^ ");\n" ^ temp4 ^ temp6;;

let declare_tensor_in_C_without_first_line name_in content_in=
  let temp1 = (List.concat (build_tensor
    (get_tensor_blocks ("[" ^ content_in ^ "]")))) in
  let temp3 = List.fold_left (fun x1 x2 -> x1 ^ x2 ^ ");\nTENLAB_assign(&" ^
    name_in ^ ",") ("TENLAB_assign(&" ^ name_in ^ ",")
    ( (Str.split (Str.regexp "[^0-9]+") content_in)) in
  let temp4 = String.sub temp3 0
    ((String.length temp3)-17-(String.length name_in)) in
  let temp5 = List.fold_left (fun x1 x2 -> x1 ^ (string_of_int x2) ^
    ");\nTENLAB_add_shape(&" ^ name_in ^ ",")
    ("\nTENLAB_add_shape(&" ^ name_in ^ ",") temp1 in
  let temp6 = String.sub temp5 0
    ((String.length temp5)-19-(String.length name_in)) in
  [DirectOut ((*"TENLAB_Tensor_create(&" ^ name_in ^ ");\n" ^*) temp4 ^ temp6)];;

(*
  "double *" ^ name_in ^ ";\n" ^ name_in ^ " =
  (double["  ^ string_of_int temp2 ^ "]) {" ^ temp4 ^ "};\nint *TENLAB_" ^
  name_in ^"_size;\nTENLAB_" ^ name_in ^ "_size =
  (int[" ^ string_of_int (List.length temp1) ^ "]) {" ^ temp6 ^ "};\n"
*)

(* Temporary Statement Constructors: *)

(*let declare_tensor_in_C name_in content_in =
  let temp1 = (List.concat (build_tensor (get_tensor_blocks
    ("[" ^ content_in ^ "]")))) in
  let temp2 = (List.fold_left (fun x1 x2 -> x1 * x2) 1 temp1) in
  let temp3 = List.fold_left (fun x1 x2 -> x1 ^ "," ^ x2) ""
    ((Str.split (Str.regexp "[^0-9]+") content_in)) in
  let temp4 = String.sub temp3 1 ((String.length temp3)-1) in
  let temp5 = List.fold_left (fun x1 x2 -> x1 ^ "," ^
    (string_of_int x2)) "" temp1 in
  let temp6 = String.sub temp5 1 ((String.length temp5)-1) in
  "double *" ^ name_in ^ ";\n" ^ name_in ^ " =
  (double["  ^ string_of_int temp2 ^ "]) {" ^ temp4 ^ "};\nint *TENLAB_" ^
  name_in ^"_size;\nTENLAB_" ^ name_in ^ "_size = (int[" ^ string_of_int
  (List.length temp1) ^ "]) {" ^ temp6 ^ "};\n";;*)


(* Deprecated: Memory Management via Compound Literals: For Future Use *)
```

```

(*
let declare_tensor_in_C name_in content_in =
  let temp1 = (List.concat (build_tensor
    (get_tensor_blocks ("[" ^ content_in ^ "]")))) in
  let temp2 = (List.fold_left (fun x1 x2 -> x1 * x2) 1 temp1) in
  let temp3 = List.fold_left (fun x1 x2 -> x1 ^ "," ^ x2) ""
    ( (Str.split (Str.regexp "[^0-9]+") content_in)) in
  let temp4 = String.sub temp3 1 ((String.length temp3)-1) in
  let temp5 = List.fold_left
    (fun x1 x2 -> x1 ^ "," ^ (string_of_int x2)) "" temp1 in
  let temp6 = String.sub temp5 1 ((String.length temp5)-1) in
  "double *" ^ name_in ^ ";\n" ^ name_in ^
    " = (double["  ^ string_of_int temp2 ^
    "]) {" ^ temp4 ^ "};\nint *TENLAB_" ^ name_in ^"_size;\nTENLAB_" ^
    name_in ^ "_size = (int[" ^
    string_of_int (List.length temp1) ^ "]) {" ^ temp6 ^ "};\n";;

let declare_tensor_in_C_without_first_line name_in content_in =
  let temp1 = (List.concat (build_tensor
    (get_tensor_blocks ("[" ^ content_in ^ "]")))) in
  let temp2 = (List.fold_left (fun x1 x2 -> x1 * x2) 1 temp1) in
  let temp3 = List.fold_left (fun x1 x2 -> x1 ^ "," ^ x2) ""
    ( (Str.split (Str.regexp "[^0-9]+") content_in)) in
  let temp4 = String.sub temp3 1 ((String.length temp3)-1) in
  let temp5 = List.fold_left (fun x1 x2 -> x1 ^ "," ^
    (string_of_int x2)) "" temp1 in
  let temp6 = String.sub temp5 1 ((String.length temp5)-1) in
  let output_string = name_in ^ " = (double["  ^
    string_of_int temp2 ^ "]) {" ^ temp4 ^ "};\nTENLAB_" ^
    name_in ^ "_size = (int[" ^ string_of_int (List.length temp1) ^
    "]) {" ^ temp6 ^ "};\n" in
  [DirectOut output_string];;
*)

let declare_for_loop_in_C name_a_in name_b_in =
  "int TENLAB_for_" ^ name_a_in ^ ";\nfor(TENLAB_for_" ^ name_a_in ^
  "=1;TENLAB_for_" ^ name_a_in ^ "<=" ^ name_b_in ^
  ".cur_content_length;TENLAB_for_" ^ name_a_in ^ "++){\n";;

let declare_arbitrary_for_loop_in_C name_a_in name_b_in =
  [DirectOut ("int TENLAB_for_" ^ name_a_in ^ ";\nfor(TENLAB_for_" ^
  name_a_in ^ "=1;TENLAB_for_"^ name_a_in ^"<=" )] @ name_b_in @
  [DirectOut (";TENLAB_for_" ^ name_a_in ^ "++){\n")];;


let list_dot_product = List.fold_left2 (fun s x y -> s + x * y) 0;;

(*let tensor_operation_abstract e1 e2 op_type = [TensorOpt [e1,e2,op_type]];;*)

let tensor_operation_constructor e1 e2 e3 op_type=
  [DirectOut (declare_tensor_in_C "TENLAB_temp_sum_a" e2)] @
```

```
  [DirectOut (declare_tensor_in_C "TENLAB_temp_sum_b" e3)] @
  [DirectOut ("double *" ^ e1 ^ ";")] @
  [DirectOut (declare_for_loop_in_C"TENLAB_inner" " TENLAB_temp_sum_a")] @
  [DirectOut ("*" ^ e1 ^
  "[TENLAB_for_TENLAB_inner] = TENLAB_temp_sum_a[ TENLAB_for_TENLAB_inner] " ^
  op_type ^ " TENLAB_temp_sum_b[TENLAB_for_TENLAB_inner];")] @ [ Curlend];;


(* let get_single_element_from_tensor_in_C name_a_in name_b_in =
  let temp1 = get_tensor_elements name_b_in in
  let temp2 = in
  "double " ^ name_a_in ^ ";\n" ^ name_a_in ^ "=" ^   *)

(* For Debugging:
get_tensor_elements "[[[5,4,3,7],[1,2,3,4]]]";;
get_tensor_blocks "[[[5,4,3,7],[1,2,3,4]]]";;
let result = build_tensor (get_tensor_blocks " [[[5,4,3,7],[1,2,3,4]]]") in
  (List.concat result);;
*)

(*
Alternate Parser Module:
tensor_lists:
  tensor_list                   { $1 }
    | tensor_lists COMMA tensor_list { (($1)) ^ "," ^ (($3)) }
    | LNPAREN tensor_lists RNPAREN { "[" ^ (($2))  ^ "]" }

tensor_list:
  LNPAREN tensor_element_list RNPAREN  { "[" ^ (String.concat "" ($2)) ^ "]" }

tensor_element_list:
  LITERAL                     { [(string_of_int $1)] }
  tensor_element_list COMMA LITERAL
    { List.rev ((string_of_int $3) :: ("," :: $1)) }
*)

(*
Symbol table: Information about all the names in scope
*)
type env = {
    function_index : int StringMap.t;
    global_index   : int StringMap.t;
    local_index    : int StringMap.t;
}

let rec enum stride n = function
    [] -> []
  | hd::tl -> (n, hd) :: enum stride (n+stride) tl

let string_map_pairs map pairs =
  List.fold_left (fun m (i, n) -> StringMap.add n i m) map pairs


(*
```

```
Translate a program in AST form into a C program in blocks. Throw an exception
if something is wrong, e.g., a reference to an unknown variable or function
*)
let translate (globals, functions) main_type=

  (* Allocate "addresses" for each global variable *)
  let global_indexes = string_map_pairs StringMap.empty (enum 1 0 globals) in

  (*
  Assign indexes to special function names; built-in "print" is special
  *)
  let built_in_functions = StringMap.add "print" (-1) StringMap.empty in
  let built_in_functions = StringMap.add "input" (-2) built_in_functions in
  let built_in_functions = StringMap.add "output" (-3) built_in_functions in
  let built_in_functions = StringMap.add "pop" (-4) built_in_functions in
  let built_in_functions = StringMap.add "dequeue" (-5) built_in_functions in
  let built_in_functions = StringMap.add "reshape" (-6) built_in_functions in
  let built_in_functions = StringMap.add "shape" (-7) built_in_functions in
  let built_in_functions = StringMap.add "set" (-8) built_in_functions in
  let built_in_functions = StringMap.add "length" (-9) built_in_functions in
  let function_indexes = string_map_pairs built_in_functions
    (enum 1 1 (List.map (fun f -> f.fname) functions)) in

  (*
  Translate into C, keeping track of the edge cases
  *)
  let translate env fdecl =
  (*
  Bookkeeping: Get number of inputs
  *)
    let num_formals = List.length fdecl.formals
    and local_offsets = enum 1 1 fdecl.locals
    and formal_offsets = enum (-1) (-2) fdecl.formals in
    let env = { env with local_index = string_map_pairs
            StringMap.empty (local_offsets @ formal_offsets) } in
  (*
  Reorder the Assignment Expressions in Intermediate Representation via
  Temporary Variables and Generate the C Code
  *)
  let expr_to_tensor_op(listin)=
    let rec env_build(listin, temporary_declarations, construction,
      finaldeclaration,tempmemorycollector,declarationtype,
      declarationsize,memoryimprint,varname)=
```

```
      match listin with
        | [] -> if declarationtype = 1 then
          (List.rev temporary_declarations) @
          (memoryimprint) @
          (List.rev tempmemorycollector) else
          (List.rev temporary_declarations) @
          [DirectOut "TENLAB_assign(&"] @
          (List.rev finaldeclaration) @ [DirectOut ","] @
          (List.rev construction) @ [DirectOut ")"] @
          (List.rev tempmemorycollector)
      | head::tail ->
        match (head) with
            TensorOpt (e1,e2,op_type) -> env_build(tail,
          (tensor_operation_constructor e1 e2
          (string_of_int (List.length tail)) op_type) @
          temporary_declarations,
          construction,
          finaldeclaration,tempmemorycollector,1,[DirectOut (e1)
            ],
          memoryimprint,varname)
        | TensorSub (e1) -> env_build(tail,
          temporary_declarations,
          construction,finaldeclaration,
        tempmemorycollector,1, [TempGetSize (List.length tail)],
        (declare_tensor_in_C_without_first_line (varname) e1),
           varname)
        | VarDeclare (e1) ->
            env_build(tail,
            temporary_declarations, construction,
            [DirectOut (e1)] @
            finaldeclaration,tempmemorycollector,declarationtype
              ,
            declarationsize,memoryimprint,e1)
        | _ -> env_build(tail,temporary_declarations,[head] @
            construction,finaldeclaration,tempmemorycollector,
          declarationtype,declarationsize,memoryimprint,varname)
      in env_build(listin, [], [], [],[], 0,
       [DirectOut "(int[1]){1};\n"],[],"")
(*
Match the expressions that don't require temporary variable
assignments in order to work
*)
    in let rec expr = function
      Literal i -> [Lit i]
    | GetTensorElement (i,j) -> [DirectOut i]
      | Id s ->
    (try [DirectOut (List.find (fun x -> x = s) (fdecl.locals))]
            with Not_found ->
              (try [DirectOut (List.find (fun x -> x = s)  fdecl
                  .formals)]
            with Not_found ->
              (try [DirectOut (List.find (fun x -> x = s)
                  globals)]
            with Not_found ->
              raise (Failure ("TENLAB Error: Undeclared tensor "
                ^ s)))))
    | TensorGet (x) ->  [TensorSub (x)]
      | Binop (e1, op, e2) -> (match e1, e2 with
          | TensorGet x,TensorGet y -> [TensorOpt (x,y,
```

```
              string_of_stmt (Bin op))]
        | Literal x,Literal y -> [Lit x]  @ [Bin op] @ [Lit y]
        | Id x,Literal y -> [DirectOut (x ^ ".Content[" ^ x ^
          ".cur_content_length-1]")] @ [Bin op] @ [Lit y]
        | Literal x,Id y -> [Lit x] @ [Bin op] @ [DirectOut y] @
            (try [DirectOut (".Content[" ^ (List.find (fun d -> d
                = y)  fdecl.formals)^ ".cur_content_length-1]")]
             with Not_found ->
               (try [DirectOut (".Content[" ^ (List.find (fun d
                  -> d = y)  globals)^ ".cur_content_length-1]")
                  ]
             with Not_found ->
               raise (Failure ("TENLAB Error: Undeclared tensor "
                  ^ y))))
             (*[DirectOut (y ^ ".Content[" ^ y ^ ".
                cur_content_length-1]")]*)
        | Id x,Id y -> [DirectOut x] @
            (try [DirectOut (".Content[" ^ (List.find (fun d -> d
                = x)  fdecl.formals)^ ".cur_content_length-1]")]
             with Not_found ->
               (try [DirectOut (".Content[" ^ (List.find (fun d
                  -> d = x)  globals)^ ".cur_content_length-1]")
                  ]
             with Not_found ->
               raise (Failure ("TENLAB Error: Undeclared tensor "
                  ^ x)))) @
          [Bin op] @ [DirectOut y] @
            (try [DirectOut (".Content[" ^ (List.find (fun d -> d
                = y)  fdecl.formals)^ ".cur_content_length-1]")]
             with Not_found ->
               (try [DirectOut (".Content[" ^ (List.find (fun d
                  -> d = y)  globals)^ ".cur_content_length-1]")
                  ]
             with Not_found ->
               raise (Failure ("TENLAB Error: Undeclared tensor "
                  ^ y))))
        | _, _ -> (expr e1) @ [Bin op] @ (expr e2))
        | Assign (s, e) ->
          (try (*let mini_env = [] in *)
          let expr_list = expr_to_tensor_op ([VarDeclare s] @ (
              expr e)) in
          (*[DirectOut ((List.find (fun x -> x = s) (fdecl.formals
              )) ^
          " = ")]*) (expr_list)
          with Not_found ->
                try [VarAssign (StringMap.find s env.global_index)
                    ] @ expr e
          with Not_found ->
                 raise (Failure ("TENLAB Error: Undefined tensor "
                    ^ s)))
      | Call ("print", actuals) -> if (List.length actuals) = 1
          then
          [(FunctionCall ("print")) ] @
          (List.concat (List.map expr (List.rev actuals))) @
          [Parend]  (* @ [DirectOut "printf(\"\\n\")"]*) else
        raise (Failure ("TENLAB Error: function print demands a
            single input."))
      | Call ("input", actuals) -> if main_type=0 then
        raise (Failure ("TENLAB Error: Inputs are only possible in
```

```
          MEX mode."))
      else if (List.length actuals) = 2 then
        [(FunctionCall ("input")) ] @
      ((List.tl(List.rev(List.fold_left(fun i j -> i @
          j @ [DirectOut ","]) [] (List.map expr (List.rev
              actuals)))))) @
          [DirectOut "-1"] @
          [Parend] else
      raise (Failure ("TENLAB Error: function input demands two
          inputs."))
    | Call ("output", actuals) ->  if main_type=0 then
      raise (Failure ("TENLAB Error: Outputs are only possible
          in MEX mode."))
      else if (List.length actuals) = 2 then
        [(FunctionCall ("output")) ] @
      ((List.tl(List.rev(List.fold_left(fun i j -> i @
          j @ [DirectOut ","]) [] (List.map expr (List.rev
              actuals)))))) @
          [DirectOut "-1"] @
          [Parend] else
      raise (Failure ("TENLAB Error: function output demands two
          inputs."))
    | Call ("length", actuals) -> if (List.length actuals) = 1
      then
        [(FunctionCall ("length")) ] @
      ((List.tl(List.rev(List.fold_left(fun i j -> i @
          j @ [DirectOut ","]) [] (List.map expr (List.rev
              actuals)))))) @
      [Parend] else
      raise (Failure ("TENLAB Error: function length demands a
          single input."))
    | Call ("pop", actuals) -> if (List.length actuals) = 1 then
        [(FunctionCall ("pop")) ] @
      ((List.tl(List.rev(List.fold_left(fun i j -> i @
          j @ [DirectOut ","]) [] (List.map expr (List.rev
              actuals)))))) @
      [Parend] else
      raise (Failure ("TENLAB Error: function pop demands a
          single input."))
    | Call ("dequeue", actuals) -> if (List.length actuals) = 1
      then
        [(FunctionCall ("dequeue")) ] @
      ((List.tl(List.rev(List.fold_left(fun i j -> i @
          j @ [DirectOut ","]) [] (List.map expr (List.rev
              actuals)))))) @
      [Parend] else
      raise (Failure ("TENLAB Error: function dequeue demands a
          single input."))
    | Call ("reshape", actuals) -> if (List.length actuals) = 2
      then
        [(FunctionCall ("reshape")) ] @
      ((List.tl(List.rev(List.fold_left(fun i j -> i @
          j @ [DirectOut ",&"]) [] (List.map expr (List.rev
              actuals)))))) @
      [Parend] else
      raise (Failure ("TENLAB Error: function reshape demands
          two inputs."))
    | Call ("set", actuals) -> if (List.length actuals) = 3 then
        [(FunctionCall ("set")) ] @
```

```
      ((List.tl(List.rev(List.fold_left(fun i j -> i @
          j @ [DirectOut ",&"]) [] (List.map expr (List.rev actuals)))))) @
      [Parend] else
      raise (Failure ("TENLAB Error: function set demands three inputs."))
    | Call (fname, actuals) ->

          [FunctionCall (* (StringMap.find fname env.function_index) *)
      fname ]  @
            (* (List.concat (List.map expr (List.rev actuals))) *)
            ((List.tl(List.rev(List.fold_left(fun i j -> i @
          j @ [DirectOut ","]) [] (List.map expr (List.rev actuals)))))) @
      [Parend]
    | Noexpr -> []

    in let rec stmt = function
      Block sl ->  List.concat (List.map stmt sl)
    | Expr e -> expr e @ [Lineend]
    | BuildTensorProd (a,b,c,d,e,f,g,h) ->
        [DirectOut (create_tensor_product_code a b c d e f g h)]
    | CreateTensor (s, e) -> [DirectOut (declare_tensor_in_C s e)]
      (* | Return e      -> expr e @ [ReturnValue num_formals] *)
    | Return e -> [ReturnValue num_formals] @ expr e @ [Lineend]
    | Clear e -> [ClearValue e] @ [Lineend]
    | Clean e -> [CleanValue e] @ [Lineend]
    | If (p, t, f) -> let t' = stmt t and f' = stmt f in
  (* expr p @ [Beq(2 + List.length t')] @
  t' @ [Bra(1 + List.length f')] @ f' *)
        [Beginf] @  expr p @ [Parend] @ [Curlbegin] @ t' @
        [DirectOut "\n}\nelse \n{\n"] @ f' @  [DirectOut "}\n"]
      | For (e1, e2, b) ->
    let b' = stmt b in
      (*[DirectOut ("TENLAB_Tensor " ^ e1 ^
    ";\nTENLAB_Tensor_create(&" ^ e1 ^ ");\n")] @ *)
    [DirectOut (declare_for_loop_in_C e1 e2)] @
    [DirectOut ("TENLAB_assign(&" ^ e1 ^ "," ^ e2 ^
    ".Content[TENLAB_for_" ^ e1 ^ "-1]);\n")] @ (b') @ [Curlend] (*@
    [DirectOut ("TENLAB_Tensor_destroy(&" ^ e1 ^ ");\n")]*)
    (*[DirectOut (declare_for_loop_in_C e1 e2)] @ (stmt b) @ [Curlend]*)
      | While (e, b) ->
    let b' = stmt b and e' = expr e in let out= match (e') with
        [DirectOut s] -> [DirectOut "while ("] @ [DirectOut s] @
      [DirectOut (".Content[" ^ s ^ ".cur_content_length-1]>0){\n") ] @
      b' @ [Curlend]
    | _ -> raise (Failure
        ("TENLAB Error: While loops take in a single tensor as argument."))
    in out

  in if (fdecl.fname)="main" then
    (*
```

```
   Generation of C Functions Compatible with the C Library
   *)
   if main_type=0 then
     [FunctionCall
   (fdecl.fname)] @
   [DirectOut
   (str_crop_last_char (List.fold_left (fun i j -> i  ^ j ^ ",")
   "" fdecl.formals))] @[Parend] @ [Curlbegin] @
   (List.fold_left (fun i j -> i @
   [DirectOut ("TENLAB_Tensor " ^ j ^ ";\n" ^
   "TENLAB_Tensor_create(&" ^ j ^ ");\n")]) [] (globals)) @
   stmt (Block fdecl.body) @ [Curlend] else
   (* Generation of Mex Code for C *)
   [DirectOut
"void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])"]
  @ [Curlbegin] @
     (List.fold_left (fun i j -> i @
     [DirectOut ("TENLAB_Tensor " ^ j ^ ";\n" ^
     "TENLAB_Tensor_create(&" ^ j ^ ");\n")]  ) [] (globals)) @
     stmt (Block fdecl.body) @ [Curlend]
 else
   [DirectOut "TENLAB_Tensor "] @
   [FunctionCall
   (fdecl.fname)] @
   [DirectOut
   (str_crop_last_char (List.fold_left
   (fun i j -> i  ^ "TENLAB_Tensor " ^ j ^ ",") "" fdecl.formals))] @
   [Parend] @ [Curlbegin] @
   (List.fold_left (fun i j -> i @
   [DirectOut ("TENLAB_Tensor " ^ j ^ ";\n" ^
   "TENLAB_Tensor_create(&" ^ j ^ ");\n")]) [] (fdecl.locals)) @
   stmt (Block fdecl.body) @ [Curlend]

   in let env = { function_index = function_indexes;
                  global_index = global_indexes;
                  local_index = StringMap.empty } in

 let entry_function_actual = [] in

 (*
 Compile the functions
 *)
 let func_bodies =
   entry_function_actual :: List.map (translate env) functions in

 { num_globals = List.length globals;
   (*
   Concatenate the compiled functions and replace the function indexes with their actual names
   *)
   text = Array.of_list (List.map (function
   _ as s -> s) (List.concat (List.rev func_bodies)))
 }
```

TENLAB Source File 5: testlab.sh

```
# Testlab: The TENLAB Testing Environment for the C Targets
# Author: Mehmet Kerem Turkcan
# Based on the MICROC code

TENLAB="./tenlab"
ulimit -t 30
globallog=testall.log
rm -f $globallog
error=0
globalerror=0
keep=0
Usage() {
    echo "Usage: testall.sh [options] [.ten files]"
    echo "-k    Keep intermediate files"
    echo "-h    Print this help"
    exit 1
}
SignalError() {
    if [ $error -eq 0 ] ; then
  echo "FAILED"
  error=1
    fi
    echo "  $1"
}
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
  SignalError "$1 differs"
  echo "FAILED $1 differs from $2" 1>&2
    }
}
Run() {
    echo $* 1>&2
    eval $* || {
  SignalError "$1 failed on $*"
  return 1
    }
}
RunFail() {
    echo $* 1>&2
    eval $* && {
  SignalError "failed: $* did not report an error"
  return 1
    }
    return 0
}
Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                             s/.ten//'`
    reffile=`echo $1 | sed 's/.ten$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."
    echo -n "$basename  "
    echo 1>&2
    echo "###### Testing $basename" 1>&2
    generatedfiles=""
```

```
    generatedfiles="$generatedfiles ${basename}.c.out" &&
    Run "$TENLAB" "-c" "<" $1 ">" ${basename}-c.c &&
    Run "gcc " ${basename}-c.c " -o " ${basename}-c.exe &&
    Run "./${basename}-c" "> ${basename}.c.out" &&
    Compare ${basename}.c.out ${reffile}.out ${basename}.c.diff
    if [ $error -eq 0 ] ; then
    echo "Works!"
    echo "###### SUCCESS" 1>&2
    else
    echo "###### Failed Horribly!" 1>&2
    globalerror=$error
    fi
}
CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                              s/.ten//'`
    reffile=`echo $1 | sed 's/.ten$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."
    echo -n "$basename "
    echo 1>&2
    echo "###### Testing $basename" 1>&2
    generatedfiles=""
    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
    RunFail "$TENLAB" "-c" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
    Compare ${basename}.err ${reffile}.err ${basename}.diff
    if [ $error -eq 0 ] ; then
    echo "Works!"
    echo "###### SUCCESS" 1>&2
    else
    echo "###### Failed Horribly!" 1>&2
    globalerror=$error
    fi
}
while getopts kdpsh c; do
    case $c in
  k) # Keep intermediate files
      keep=1
      ;;
  h) # Help
      Usage
      ;;
    esac
done
shift `expr $OPTIND - 1`
if [ $# -ge 1 ]
then
    files=$@
else
    files="tests/fail-*.ten tests/test-*.ten"
fi
for file in $files
do
    case $file in
  *test-*)
      Check $file 2>> $globallog
      ;;
```

```
  *fail-*)
      CheckFail $file 2>> $globallog
      ;;
  *)
      echo "unknown file type $file"
      globalerror=1
      ;;
    esac
done
exit $globalerror
```

TENLAB Source File 6: testc.sh

```
# Testlab: The TENLAB Testing Environment for the C Targets
# Author: Mehmet Kerem Turkcan
# Based on the MICROC code

TENLAB="./tenlab"
ulimit -t 30
globallog=testall_c.log
rm -f $globallog
error=0
globalerror=0
keep=0
Usage() {
    exit 1
}
SignalError() {
    if [ $error -eq 0 ] ; then
  echo "FAILED"
  error=1
    fi
    echo "  $1"
}
Compare() {
    generatedfiles="$generatedfiles $3"
    echo diff -b $1 $2 ">" $3 1>&2
    diff -b "$1" "$2" > "$3" 2>&1 || {
  SignalError "$1 differs"
  echo "FAILED $1 differs from $2" 1>&2
    }
}
Run() {
    echo $* 1>&2
    eval $* || {
  SignalError "$1 failed on $*"
  return 1
    }
}
RunFail() {
    echo $* 1>&2
    eval $* && {
  SignalError "failed: $* did not report an error"
  return 1
    }
    return 0
}
Check() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                             s/.c//'`
    reffile=`echo $1 | sed 's/.c$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."
    echo -n "$basename..."
    echo 1>&2
    echo "###### Testing $basename" 1>&2
    generatedfiles=""
    generatedfiles="$generatedfiles ${basename}.c.out" &&
    Run "gcc ctests/${basename}.c" " -o " ${basename}-c.exe &&
    Run "./${basename}-c" "> ${basename}.c.out" &&
```

```
    Compare ${basename}.c.out ${reffile}.out ${basename}.cc.diff
    if [ $error -eq 0 ] ; then
    echo "Works!"
    echo "###### Failed Horribly!" 1>&2
    else
    echo "###### FAILED" 1>&2
    globalerror=$error
    fi
}
CheckFail() {
    error=0
    basename=`echo $1 | sed 's/.*\\///
                             s/.ten//'`
    reffile=`echo $1 | sed 's/.ten$//'`
    basedir="`echo $1 | sed 's/\/[^\/]*$//'`/."
    echo -n "$basename..."
    echo 1>&2
    echo "###### Testing $basename" 1>&2
    generatedfiles=""
    generatedfiles="$generatedfiles ${basename}.err ${basename}.diff" &&
    RunFail "$TENLAB" "-b" "<" $1 "2>" "${basename}.err" ">>" $globallog &&
    Compare ${basename}.err ${reffile}.err ${basename}.diff
    if [ $error -eq 0 ] ; then
    echo "Works!"
    echo "###### Failed Horribly!" 1>&2
    else
    echo "###### FAILED" 1>&2
    globalerror=$error
    fi
}
shift `expr $OPTIND - 1`
files="ctests/test-*.c"
for file in $files
do
    case $file in
  *test-*)
      Check $file 2>> $globallog
      ;;
  *)
      echo "unknown file type $file"
      globalerror=1
      ;;
    esac
done
exit $globalerror
```

TENLAB Source File 7: tenlab_preamble.c

```c
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>


typedef struct TENLAB_Tensor
{
 int *Shape;
 double *Content;
 size_t max_content_length;
 size_t cur_content_length;
 size_t shape_length;
} TENLAB_Tensor;

void TENLAB_Terminate()
{
 exit(1);
}

void TENLAB_add_element(TENLAB_Tensor *X, double y)
{
  if(X->cur_content_length == X->max_content_length)
  {
    int new_max_content_length = X->max_content_length + X->Shape[X->shape_length-1];
  //Different OS's may have SIZE_T_MAX instead
    if((new_max_content_length > X->max_content_length) && (new_max_content_length < SIZE_MAX / sizeof(double)))
    {
      double *new_Content = (double*) realloc(X->Content, new_max_content_length * sizeof(double));
      if(new_Content != NULL)
      {
    X->Content = new_Content;
    X->max_content_length = new_max_content_length;
      }
      else
      {
    printf("\n TENLAB Error: Memory filled during size reallocation.");
    TENLAB_Terminate();
   }
    }
    else
    {
   printf("\n TENLAB Error: Memory overflow.");
   TENLAB_Terminate();
  }
  }
  X->Content[X->cur_content_length] = (double) y;
  X->Shape[0] = X->Shape[0]+1;
  X->cur_content_length++;
}

```

```
void TENLAB_add_length(TENLAB_Tensor *X)
{
  if(X->cur_content_length == X->max_content_length)
  {
    int new_max_content_length = X->max_content_length + X->Shape[X->shape_length-1];
  //Different OS's may have SIZE_T_MAX instead
    if((new_max_content_length > X->max_content_length) && (new_max_content_length < SIZE_MAX / sizeof(double)))
    {
      double *new_Content = (double*) realloc(X->Content, new_max_content_length * sizeof(double));
      if(new_Content != NULL)
      {
    X->Content = new_Content;
    X->max_content_length = new_max_content_length;
      }
      else
      {
    printf("\n TENLAB Error: Memory filled during size reallocation.");
    TENLAB_Terminate();
   }
    }
    else
    {
   printf("\n TENLAB Error: Memory overflow.");
   TENLAB_Terminate();
  }
  }
  X->Content[X->cur_content_length] = (double) X->cur_content_length;
  X->Shape[0] = X->Shape[0]+1;
  X->cur_content_length++;
}

void TENLAB_pop_element(TENLAB_Tensor *X)
{
  if(X->cur_content_length > 0 )
  {
    int new_max_content_length = X->max_content_length - 1;
  //Different OS's may have SIZE_T_MAX instead
    if((new_max_content_length < SIZE_MAX / sizeof(double)))
    {
      double *new_Content = (double*) realloc(X->Content, new_max_content_length * sizeof(double));
      if(new_Content != NULL)
      {
    X->Content = new_Content;
    X->max_content_length = new_max_content_length;
      }
      else
      {
    printf("\n TENLAB Error: Memory filled during element removal.");
    TENLAB_Terminate();
   }
    }
    else
```

```
    {
   printf("\n TENLAB Error: Memory overflow.");
   TENLAB_Terminate();
  }
   if(X->Shape[0] > 1 )
    X->Shape[0] = X->Shape[0]-1;
   X->cur_content_length--;
  }
}



void TENLAB_add_shape(TENLAB_Tensor *X, double y)
{
 int new_max_shape = X->shape_length + 1;
 //Different OS's may have SIZE_T_MAX instead
 if (new_max_shape < SIZE_MAX / sizeof(int))
 {
  int *new_Shape = (int*) realloc(X->Shape, new_max_shape * sizeof(int));
  if(new_Shape != NULL)
  {
   X->Shape = new_Shape;
  }
  else
  {
   printf("\n TENLAB Error: Memory filled during size reallocation.");
   TENLAB_Terminate();
  }
 }
 else
 {
  printf("\n TENLAB Error: Memory overflow.");
  TENLAB_Terminate();
 }
 X->Shape[X->shape_length] = (int) y;
 X->shape_length++;
}

void TENLAB_add_element_at_specific_position(TENLAB_Tensor *X, TENLAB_Tensor *Y, TENLAB_Tensor *Z)
{
 if (Y->cur_content_length==X->shape_length)
 {
  int adding_index=1;
  for (int i=0;i<X->shape_length;i++)
   adding_index=adding_index * Y->Content[i];
  adding_index=(int)adding_index;
  if(X->cur_content_length < adding_index-1)
  {
   int new_max_content_length = adding_index-1;
   if((new_max_content_length > X->max_content_length) && (new_max_content_length < SIZE_MAX / sizeof(double)))
   {
    double *new_Content = (double*) realloc(X->Content, new_max_content_length * sizeof(double));
    if(new_Content != NULL)
    {
```

```
     X->Content = new_Content;
     X->max_content_length = new_max_content_length;
     for (int i=X->cur_content_length;i<=adding_index-1;i++)
      X->Content[i]=0;
     X->cur_content_length=adding_index-1;
    }
    else
    {
     printf("\n TENLAB Error: Memory filled during size reallocation.");
     TENLAB_Terminate();
    }
   }
   else if (new_max_content_length >= SIZE_MAX / sizeof(double))
   {
    //printf("\n TENLAB Error: Memory overflow.");
    //TENLAB_Terminate();
   }
  }
  X->Content[adding_index-1] = Z->Content[Z->cur_content_length-1];
 }
 else
 {
  printf("\n TENLAB Error: Assignment dimension does not match.");
  TENLAB_Terminate();
 }
}

void TENLAB_add_element_at_linear_index(TENLAB_Tensor *X,int adding_index, double y)
{
 if (X->cur_content_length==X->shape_length)
 {
  if(X->cur_content_length < adding_index-1)
  {
   int new_max_content_length = adding_index-1;
   if((new_max_content_length > X->max_content_length) && (new_max_content_length < SIZE_MAX / sizeof(double)))
   {
    double *new_Content = (double*) realloc(X->Content, new_max_content_length * sizeof(double));
    if(new_Content != NULL)
    {
     X->Content = new_Content;
     X->max_content_length = new_max_content_length;
     for (int i=X->cur_content_length;i<=adding_index-1;i++)
      X->Content[i]=0;
     X->cur_content_length=adding_index-1;
    }
    else
    {
     printf("\n TENLAB Error: Memory filled during size reallocation.");
     TENLAB_Terminate();
    }
   }
   else if (new_max_content_length >= SIZE_MAX / sizeof(double))
```

```
    {
     //printf("\n TENLAB Error: Memory overflow.");
     //TENLAB_Terminate();
    }
   }
   X->Content[adding_index-1] = y;
  }
  else
  {
   printf("\n TENLAB Error: Assignment dimension does not match."
       );
   TENLAB_Terminate();
  }
}


void TENLAB_Tensor_create(TENLAB_Tensor *X)
{
   X->Shape = NULL;
   X->Content = NULL;
   X->shape_length = 1;
  int *new_Shape = (int*) realloc(X->Shape, (X->shape_length) *
      sizeof(int));
  if(new_Shape != NULL)
   {
     X->Shape = new_Shape;
   X->Shape[0] = 1;
   }
  else
  {
   printf("\n TENLAB Error: Memory filled during tensor
       initialization.");
   TENLAB_Terminate();
  }
  X->cur_content_length = 0;
  X->max_content_length = 0;
}

void TENLAB_Tensor_destroy(TENLAB_Tensor *X)
{
   free(X->Content);
  free(X->Shape);
   X->max_content_length = 0;
   X->cur_content_length = 0;
  X->shape_length = 0;
}

void TENLAB_Tensor_duplicate(TENLAB_Tensor *Y,TENLAB_Tensor *X)
{
   free(Y->Content);
  free(Y->Shape);
   Y->shape_length = X->shape_length;
  Y->cur_content_length = X->cur_content_length;
  Y->max_content_length = X->max_content_length;
  double *new_Content = (double*) malloc(X->max_content_length *
      sizeof(double));
  if(new_Content != NULL)
  {
   Y->Content = new_Content;
```

```
  memcpy(Y->Content,X->Content,X->max_content_length * sizeof(double));
 }
 else
 {
  printf("\n TENLAB Error: Memory filled during tensor duplication.");
  TENLAB_Terminate();
 }
 int *new_Shape = (int*) malloc( X->shape_length * sizeof(int));
 if(new_Shape != NULL)
 {
  Y->Shape = new_Shape;
  memcpy(Y->Shape,X->Shape,X->shape_length * sizeof(int));
 }
 else
 {
  printf("\n TENLAB Error: Memory filled during tensor duplication.");
  TENLAB_Terminate();
 }
}

void TENLAB_Tensor_reshape(TENLAB_Tensor *Y,TENLAB_Tensor *X)
{
 free(Y->Shape);
 Y->shape_length = X->cur_content_length;
 int *new_Shape = (int*) malloc( X->cur_content_length * sizeof(int));
 if(new_Shape != NULL)
 {
  Y->Shape = new_Shape;
  for(int i=1;i<=X->cur_content_length;i++)
   Y->Shape[i-1]=(int) round(X->Content[i-1]);
 }
 else
 {
  printf("\n TENLAB Error: Memory filled during tensor reshaping.");
  TENLAB_Terminate();
 }
}

void TENLAB_dequeue_element(TENLAB_Tensor *X)
{
  if(X->cur_content_length > 0 )
  {
    X->Content++;
    int new_max_content_length = X->max_content_length - 1;
  //Different OS's may have SIZE_T_MAX instead
    if((new_max_content_length < SIZE_MAX / sizeof(double)))
    {
      //double *new_Content = (double*) realloc(X->Content, new_max_content_length * sizeof(double));
      //if(new_Content != NULL)
      //{
       //X->Content = new_Content;
       X->max_content_length = new_max_content_length;
      //}
```

```
      //else
      //{
    //printf("\n TENLAB Error: Memory filled during element
        removal.");
    //TENLAB_Terminate();
   //}
    }
    else
    {
   printf("\n TENLAB Error: Memory overflow.");
   TENLAB_Terminate();
  }
   if(X->Shape[0] > 1 )
    X->Shape[0] = X->Shape[0]-1;
   X->cur_content_length--;
  }
}

void TENLAB_Tensor_nonpointing_duplicate(TENLAB_Tensor *Y,
    TENLAB_Tensor X)
{
  //free(Y->Content);
 //free(Y->Shape);
 TENLAB_Tensor temp;
 TENLAB_Tensor_create(&temp);
 //TENLAB_Tensor_duplicate(&temp,&X);
 //printf("This worked");
 //TENLAB_Tensor_destroy(Y);
 //TENLAB_Tensor_create(Y);
 Y->shape_length = X.shape_length;
 Y->cur_content_length = X.cur_content_length;
 Y->max_content_length = X.max_content_length;
 double *new_Content = (double*) malloc(X.max_content_length *
     sizeof(double));
 if(new_Content != NULL)
 {
  Y->Content = new_Content;
  memcpy(Y->Content,X.Content,X.max_content_length * sizeof(
      double));
 }
 else
 {
  printf("\n TENLAB Error: Memory filled during tensor
      duplication.");
  TENLAB_Terminate();
 }
 int *new_Shape = (int*) malloc( X.shape_length * sizeof(int));
 if(new_Shape != NULL)
 {
  Y->Shape = new_Shape;
  memcpy(Y->Shape,X.Shape,X.shape_length * sizeof(int));
 }
 else
 {
  printf("\n TENLAB Error: Memory filled during tensor
      duplication.");
  TENLAB_Terminate();
 }
 TENLAB_Tensor_destroy(&temp);
```

```
}

#define TENLAB_assign(a, b) _Generic(b, int: TENLAB_add_element,
     double: TENLAB_add_element, TENLAB_Tensor :
    TENLAB_Tensor_nonpointing_duplicate)(a, b)

void TENLAB_Tensor_check_size(TENLAB_Tensor *X,TENLAB_Tensor *Y)
{
 if (X->Shape[0]!=Y->Shape[0])
 {
  printf("\n TENLAB Error: Total number of dimensions are
      different.");
  TENLAB_Terminate();
 }
 else
 {
  for (int i=1;i<=X->shape_length;i++)
  {
   if (X->Shape[i-1]!=X->Shape[i-1])
   {
    printf("\n TENLAB Error: Dimensions don't match.");
    TENLAB_Terminate();
   }
  }
 }
}

void TENLAB_Tensor_force_scalar(TENLAB_Tensor *X)
{
 if (X->cur_content_length>1)
 {
  printf("\n TENLAB Error: A scalar was sought.");
  TENLAB_Terminate();
 }
}

void TENLAB_Tensor_while_is_not_scalar(TENLAB_Tensor *X)
{
 if (X->cur_content_length>1)
 {
  printf("\n TENLAB Warning: A while loop only considers the
      first element of a tensor.");
 }
}

void TENLAB_Tensor_print(TENLAB_Tensor X)
{
 if (X.cur_content_length>=1)
 {
  for(int i = 1; i <= X.cur_content_length; i++)
  {
   printf("%f\n",X.Content[i-1]);
  }
 }
}

void TENLAB_Tensor_shape_print(TENLAB_Tensor X)
{
  for(int i = 1; i <= X.shape_length; i++)
```

```
  {
    printf("%f\n",(double) X.Shape[i-1]);
  }
}

void TENLAB_Tensor_round_all(TENLAB_Tensor X)
{
 if (X.cur_content_length>=1)
 {
  for(int i = 1; i <= X.cur_content_length; i++)
  {
   X.Content[i-1]=round(X.Content[i-1]);
  }
 }
}

void TENLAB_Tensor_round(TENLAB_Tensor X)
{
 if (X.cur_content_length>=1)
 {
  X.Content[X.cur_content_length-1]=round(X.Content[X.
      cur_content_length-1]);
 }
}
```

9 Project Log

```
commit ec90b0555cf11837c76cabdc66daaf7bcec59ffe
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed May 11 19:11:11 2016 -0400

    Pushing the (probably) last set of changes.

commit 32b6835f68f9a508b07e7799c1dcb0fd5b13d7db
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed May 11 13:13:16 2016 -0400

    Added the demo.

commit 1542f10b947248edd35a503450f4ab51088462a0
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed May 11 11:12:43 2016 -0400

    I believe everything is essentially done.

commit 0c2545338d869fb8a88b29893c6e5d5646d44f83
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed May 11 09:04:41 2016 -0400

    Added some parts of the final presentation.

commit f5ca5827ba1ed46b1a99242e894466e8cc870ba6
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed May 11 00:29:01 2016 -0400

    10 new test cases. Better error handling. Done with nearly
        everything.

commit 5a925b5901e2c69ea8a065c23ea1d9842d9eff35
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 21:43:10 2016 -0400

    Satisfied with this first draft. Now let's see what we can
        do in the time we have!

commit e9e040d6c87ae7d70b6208f0165b0db495586b0a
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 20:17:59 2016 -0400

    The final stretch! Need to reimplement the tensor product
        into the LRM.

commit 72f8d01eca97d34d070292ccacdb9daa20c88c16
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 18:35:17 2016 -0400

    Close to being done. Still not satisfied with a number of
        sections.

commit 796ff1fb1be020c57d94d40c8402445628b95727
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 16:59:53 2016 -0400

```

```
    Slowly getting there in regards to the report. Tutorial,
        testing and the tensor product are all that's left.

commit b28b838b4cb9d635bc038569f9fb76f414f2c469
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 15:50:22 2016 -0400

    Still heavily working on the final report. The LRM still
        needs a lot of updates.

commit f2a751b746d2380018d6c17267b4ab5bad737048
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 13:09:28 2016 -0400

    Added a nice architecture diagram.

commit 38ce6fe4d1d0cea14bed37443923e2d574261e35
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 12:14:05 2016 -0400

    Continuing to work on the final report. Architectural Design
         section is almost complete.

commit 8339a4463218e911d208fa6d20015cbdf060954e
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 10:31:23 2016 -0400

    Still working on the final report.

commit 2909f37bb70da1f8569d79b17b80dad7d4e13614
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 08:45:42 2016 -0400

    Continuing to work on the final report.

commit cfb77cef30c88174ff7b68a78ae2688bb9ad47eb
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 07:49:20 2016 -0400

    Tensor product done. Now perhaps we should change the
        representation to the MATLAB format from the C format.

commit a04a9f0add4c8719b025f83c4cf563fef3f9fe47
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue May 10 00:33:53 2016 -0400

    Tensor product integrated. Does not appear to work.

commit 282e8c2d5b5e787031cc0aef290e053a737779a1
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 21:47:00 2016 -0400

    More additions to the final report. Continuing work on the
        tensor product.

commit 57614ed10757a099b20e7724234e924c01a0074e
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 20:08:09 2016 -0400

```

```
    Perhaps the problem is now fixed?

commit 97b3c39fdb17afa815462bb65f0c300a1749052f
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 19:13:49 2016 -0400

    Added some new test cases. There is a failure.

commit 5c607320eea1cf25e826de398ba249ca87c009f0
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 18:35:34 2016 -0400

    Some updates to the final report.

commit 23e697df8a506049dc8c431788844c6a58052880
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 16:07:53 2016 -0400

    Mex is done. Need more functions.

commit b44f257bddd861cbecf7c3b091fb505aa0638915
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 14:30:42 2016 -0400

    Mex interface is almost done.

commit ec57b4c8153f440120dfa40b62f299b1ea866a41
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 12:10:12 2016 -0400

    Library now compatible with MATLAB. No outputs yet.

commit bc73525f6688e041945969c42055d10ff6db1a11
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 11:41:54 2016 -0400

    Adding CPP control files.

commit db1df6a1d7532671d6180dd138af1305acddbe08
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 02:25:19 2016 -0400

    Improving the final report. Adding empty husks for mex integration.

commit d20a3ab4f75c6913ea512c759cd3df504d318623
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 01:46:52 2016 -0400

    More updates to the project report; beginning to integrate the LRM. Still need to work on some test cases.

commit 857204636cbf4bea9f805b70c1a0a3cf172b95e7
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon May 9 00:12:49 2016 -0400

    Updates to project report.

commit a6a47ab63d221d21ae8142a6c945f9e00d7e1aca
```

```
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 20:41:28 2016 -0400

    Style improvements.

commit 7b856908c3389a8f7b23a60520d3efcadb18adb3
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 17:57:56 2016 -0400

    Still working on tests. Also working on the final report, but not yet ready to show a draft.

commit 024e72d69bcab81bab9d7bfd509bce9a759ffe9f
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 14:37:49 2016 -0400

    Even more C tests.

commit 6190bbd797d0ef30f1a76eb217da72ed06403957
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 13:50:03 2016 -0400

    Continuing to add C test cases.

commit e2548bbccfb68a274cac8ac4dfd2a9888db474ed
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 13:21:15 2016 -0400

    New test cases. Need to add more C tests. Then focus on adding more content.

commit dee3945c24cd0d3673f5504f2e2b704071da7dde
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 11:47:35 2016 -0400

    More tests.

commit cbbdcd3d1287157ed40a2ffde666e7a56444a653
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 10:33:01 2016 -0400

    Minor alterations to the preamble.

commit 59a529c166c56d2c92e6a7c076e3b985d260c407
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 10:24:27 2016 -0400

    Completed adding a basic testing script for the C library.

commit fe733712d9fbd60c8f0182a39ffe399bd7ea64e9
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun May 8 09:38:48 2016 -0400

    Beginning to build the C testing environment.

commit 8e284d0b2abdd843bb8efec8caa472578526236b
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sat May 7 20:51:22 2016 -0400

```

```
    Beginning to update the white paper for the final report.
        Just random ideas at the moment.

commit e429fc2b6ea51464501e26d8aa8cda04aacb786d
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sat May 7 19:13:31 2016 -0400

    Another test dealt with. There are some memory problems that
         are still cropping up; need to devise tests to deal
        with them.

commit 2eb2dce46fe7bb38e64d6775198c717c2ad37bf2
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Fri May 6 22:49:15 2016 -0400

    A lot of the other test cases are running again. New test
        case for For.

commit 5d6cffc671b925c54a2229b6d3659d73612b62ed
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Fri May 6 22:39:08 2016 -0400

    Another test case integrated. For loops are quite cool now.

commit d947f5227e5407ebd4cc5a868242a0297a506132
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Fri May 6 21:27:56 2016 -0400

    One more test case down. Some parser behaviours have been
        fixed.

commit d40aa45d356459812b1d098622285fe1bc3b8f67
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Fri May 6 18:34:24 2016 -0400

    Removed a number of discrepancies and began the integrate
        the scalar/tensor operations for the users' convenience.

commit dce623cb6645708451a8ee6e6c8fd23b7dfac172
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Thu May 5 22:36:13 2016 -0400

    More fundamental functions that access the Tensor struct
        automatically. Beginning to reintegrate the tests one by
         one.

commit 356bb5f1869c9cdd793286cb38d0186f1f86a0bb
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Thu May 5 16:11:57 2016 -0400

    Added a better version of the C backend libraries for the
        language as well as some test cases for prototyping.

commit 2337a7e45769bf3b36979e22e39fac2bfcd06613
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Thu May 5 12:01:41 2016 -0400

    Added the first elements of the preamble necessary.
        Integrated with the language. Cleaned legacy code.
```

```

commit 65dfbf05dfc75b9c24c2a1a23ee4dc2208077ff3
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Thu May 5 00:03:27 2016 -0400

    Adding the prototypes for the execution-time error checking
        modules. The idea is to decouple the shape information
        from the content for certain operations in order to
        allow for more freedom to the user.

commit e1c64ea5fcb693392a714d25470dc3ab875c7611
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed Apr 27 13:26:26 2016 -0400

    Spring cleaning done. Mex interface added, but without I/O.

commit aa08782f8f81078b4e12d1bdb08f7f1c773c2a5c
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Apr 26 23:46:23 2016 -0400

    Almost done with the cleaning. Better memory handling.
        Broadcasting between scalars and tensors still has some
        glaring problems. Many problems cannot be handled by the
         compiler, but we can solve problems using indirection
        or through the introduction of a "scalar" type.

commit 85bbd19f6ba9e6cad234a6e3d9c9f9f0f6de13c5
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Apr 26 15:14:37 2016 -0400

    More cleaning. Removed the now-useless execute function
        completely. Reduced the compilation warnings; the ones
        that still exist are for code that will be added later.

commit 8aa11198e67ead4723bf9bb79c5c567b25c1c2d7
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Apr 26 12:03:59 2016 -0400

    Spring cleaning continues. Slowly removing dependencies on
        the DirectOut function.

commit ce0fe3257a1284f3d2e4ff0562feb0e10d0d54ef
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Apr 26 10:51:42 2016 -0400

    Beginning to do some spring cleaning. Also implementing
        blocks for garbage collection. Next step: replace the
        compound literals.

commit 73be3a6704018ce60c92dfe0cc8190d94a4458fc
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun Apr 24 23:38:54 2016 -0400

    Various improvements; the language is working apart from
        some scalar/tensor casts.

commit 316e9edeb1a9173b17aeecabab1e1f9e116ef6a3
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sun Apr 24 22:12:10 2016 -0400
```

```
    Fixed numerous C crashes due to dynamic memory allocation.
        Size inference is still a huge problem. Have to consider
         I/O next to focus on solving the remaining problems.

commit 093795a65d8d877b9a1d59415b53343442283d57
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sat Apr 23 13:17:40 2016 -0400

    Everything but size inference is operational. Need to
        implement checks on sizes and garbage collection.

commit 669097b2994dbf7fe123c699fcce022cb3a295fe
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Sat Apr 23 00:40:07 2016 -0400

    Vast improvements; the optimized code generator is nearly
        complete.

commit 8909fef876a148dd01bf9ec86f643a55565027cc
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Fri Apr 22 21:36:17 2016 -0400

    Creating a dev build to keep track of all the various
        changes, even though the build is non-functioning.
        Started building the optimized code generator; will be
        done soon.

    Next Steps: Implement the tensor product into this. Get the
        .mex interface done. Then consider other possible
        interfaces.

commit 8fe2ceb4fd36eaa0cfda460880e3f0d47125bb76
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed Apr 13 00:11:11 2016 -0400

    Major strides in tensor-tensor operations, multi-argument
        functions are now operational, fail cases are in and
        working for the test system, minor bugs in loop
        structures fixed. Tensor-integer operations are not yet
        working. Tensor-tensor operations rely on linear
        indexing (i..e not tracked by compiler). Tensor-tensor
        operations are not compiling. Tensor size checks for the
         tests are not in.

commit 39e33a06a109620731d87d0d4297706b838c3418
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed Apr 6 00:29:47 2016 -0400

    Replaced the tests with the new ones.

commit ad7aa0fadde59d76b4d34f0cc9a9c72bb8b77a6b
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed Apr 6 00:23:41 2016 -0400

    no message

commit d55be8a1e04251ac47605b39310872034888fdd2
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
```

```
Date:   Wed Apr 6 00:00:03 2016 -0400

    Improved Hello World files. Variable assignment is not working. Probably need to begin to write a C library backend for some nonrecursive operations.

commit c1cebe799377638514c82a5456059e024d314603
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Apr 5 15:48:47 2016 -0400

    Hello World and variants fully compilable. Working on accessing tensor elements next.

commit 4cc19843d55bea8b2f0ca726851c0cd48c70f213
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon Apr 4 23:53:25 2016 -0400

    Tensors now work! Only need to write the code to extract their dimensions as well. We need the function for the tensor product for the standard library, written in C.

commit 499c50d4f5bef937db9280f010f68fd25a05e1a7
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon Apr 4 19:10:28 2016 -0400

    Compilation works! Also added templates for tensor assignments.

commit 492f16e2cdee650a9abe9639db0a20fda4fa94c9
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon Apr 4 16:28:05 2016 -0400

    Only source.

commit 0c8fb4fbb08fe327aaf979c6fdd7c437a14f48ee
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed Mar 30 23:31:24 2016 -0400

    Basic variables are working, compiler checks are not yet broken. Now need to implement loops.

commit 7b363efefbe450d6b1396a22f61f69c39181ab09
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed Mar 30 21:03:09 2016 -0400

    Further improvements.

commit bb132acabf40bdc886ce12e5bd6ecf8647d76e0f
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed Mar 30 18:16:48 2016 -0400

    Beginnings of the compiler. Use "make" on the directory, then "bash testall.sh" for tests. Only look at the .b outputs for testall.sh. To do: C code generation.

commit 2eb44320cc232aeb6e6502a26df46c04b161a5c3
Author: ycemsubakan <csubakan@gmail.com>
Date:   Mon Mar 7 20:12:51 2016 -0500

```

```
    added star between A and B

commit 239c7b34d2b7a0169e727af1273dfee3c1f36cc0
Author: ycemsubakan <csubakan@gmail.com>
Date:   Mon Mar 7 20:11:53 2016 -0500

    added star between A and B

commit 312a568053fcce647e98d63077edbc8c1e0dd547
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Mon Mar 7 19:39:26 2016 -0500

    New manual.

commit b276a26471175212f7baf17cc87b670d467bc067
Author: ycemsubakan <csubakan@gmail.com>
Date:   Mon Mar 7 13:57:58 2016 -0500

    added an example C compilation for TP

commit 470e8789d62cd948d1e1d8d505dadb86d0c4468d
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Mar 1 18:29:35 2016 -0500

    Some further improvements to both the language manual and
        the parser/scanner. Possible errors due to the way IF is
         coded.

commit 71e16787aaf7f06a83d925ad27235a13344a5e4b
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Mar 1 16:04:09 2016 -0500

    First drafts for scanner and parser.

commit ec153aa9c8a11e7e2f5fa90321e75113a45d7420
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Feb 23 23:49:59 2016 -0500

    More traditional sections for the reference manual, may
        become important. Ignore first page; extra details that
        are possibly also useful for the meetings.

commit 9a03e517e20c00c9f76d4ab10c88a573536fe1ff
Author: ycemsubakan <csubakan@gmail.com>
Date:   Tue Feb 23 14:02:28 2016 -0500

    adding_lang_ref_folder

commit 2e612c6d247814d7b9628af2f632403272d4a7f0
Author: ycemsubakan <csubakan@gmail.com>
Date:   Tue Feb 23 13:56:44 2016 -0500

    adding_language_ref_folder

commit 296273ab02cc76121a418e3ca36fc9490b64c3be
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Feb 9 20:20:36 2016 -0500

    Lots of minor edits; take an another look before submission.
```

```

commit 40f4514ccad0e453c0c5618a4448f96dbf720eef
Author: Cem Subakan <cemsubakan@dyn-160-39-172-138.dyn.columbia.
    edu>
Date:   Wed Feb 10 00:18:54 2016 -0600

    I think this is the final version

commit 502b44201193691e4e2d7095f5525d6194d6cc64
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Tue Feb 9 01:04:58 2016 -0500

    Additions from Cem and some improvements + syntax from me.

commit dabe02777c5247da479a62a7967e7a7cfa876fdf
Author: Mehmet Kerem Turkcan <mkt2126@columbia.edu>
Date:   Wed Feb 3 19:32:01 2016 -0500

    Template?
```

References

[1] M. U. Guide, "The mathworks," *Inc., Natick, MA*, vol. 5, p. 333, 1998.

[2] R. Ihaka and R. Gentleman, "R: a language for data analysis and graphics," *Journal of computational and graphical statistics*, vol. 5, no. 3, pp. 299–314, 1996.

[3] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, "The numpy array: a structure for efficient numerical computation," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[4] F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I. Goodfellow, A. Bergeron, N. Bouchard, D. Warde-Farley, and Y. Bengio, "Theano: new features and speed improvements," *arXiv preprint arXiv:1211.5590*, 2012.

[5] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems, 2015," *Software available from tensorflow.org*.

[6] I. Markovsky, O. Debals, and L. De Lathauwer, "Sum-of-exponentials modeling and common dynamics estimation using tensorlab," in *Latent Variable Analysis and Signal Separation*, 2015, pp. 1–8.

[7] M. Grant, S. Boyd, and Y. Ye, "Cvx: Matlab software for disciplined convex programming," 2008.