CS4121 Cminus Expression Interpreter Project
Due Date: Wednesday, January 29, 2020 at *5pm*

Purpose

The purpose of this project is to gain experience in giving meaning to a programming language by interpreting a subset of Cminus. Specifically, you will interpret integer and string write operations, integer read operations, integer arithmetic and logical expressions and assignment statements.

Project Summary

In this project, you will add actions to a parser and a scanner provided by me. You must add code to do the following:

1. Design a data structure to store string constants and integer variables declared in a Cminus program and to track variable values.
2. Pass attributes of tokens from the scanner to the parser when needed.
3. Interpret string write statements.
4. Interpret integer write statements.
5. Interpret integer read statements.
6. Interpret integer expressions.
7. Interpret logic expressions.
8. Interpret assignment statements which assigns values to integer variables.

Requirements

Write all of your code in C or C++ . It will be tested on a Rekhi CS lab machine and MUST work there. You will receive no special consideration for programs which "work" elsewhere, but not on a CS machine.

Input. The file `CminusProject1.tgz` contains the parser needed to begin this project. You will need to modify the actions in the project files `parser/CminusParser.y` and `parser/CminusScaner.l` to do this project. Currently, the actions in the parser just emit the rules that are reduced. You also need to set up certain token attributes in the scanner. Sample input for this project is provided in the project directory `input`. To run your interpreter, use the command

```
cmc <file>.cm
```

You will see <file>.s which contains the output of the Cminus program. Type in your input if the program contains read statements.

Submission. Your code should be well-documented. You will submit all of your files, by tarring up your project directory using the command

```
tar -czf CminusProject1.tgz CminusProject1
```

Submit the file `CminusProject1.tgz` via the CS4121 Canvas page. Make sure you do a 'make clean' of your directory before executing the tar command. This will remove all of the '.o' files and make your tar file much smaller.

Data Structures and Documentation I have provided several C data structures for those who will be programming in C. There are doubly linked list, symbol table and string manipulation routines in the directory `CminusProject1/util`. The HTML Doxygen documentation for the provided code is in `CminusProject1/Documentation/html/index.html`. You may ask me any questions regarding these routines. You will not likely need any of these structures now, but you may want to familiarize yourself with them. For those coding in C++, you may use STL.

Makefile Structure The Makefiles for the project are set up to automatically generate make dependences. In a particular directory (*e.g.*, `parser`), you may add new files for compilation by adding the source file name to the `SRCS` variable declaration on the first line of that directory's `Makefile`. For example, to add the file `newfile.c` to be compiled in the `parser` directory, change the first line of `parser/Makefile` from

```
SRCS = CminusScanner.c CminusParser.c
```

to

```
SRCS = CminusScanner.c CminusParser.c newfile.c
```

Nothing else needs to be done. Do not add source files to the root directory `CminusProject1` as the make files assume there are no source files in that directory.

If you would like to add your own subdirectory (*e.g.*, `newdir`) to `CminusProject1`, then change the line

```
DIRS = parser util
```

in `CminusProject1/Makefile` to

```
DIRS = parser util newdir
```

and the line

```
LIBS = parser/libparser-g.a util/libutil-g.a
```

in `CminusProject1/Makefile` to

```
LIBS = parser/libparser-g.a util/libutil-g.a newdir/libnewdir-g.a
```

Then, copy `util/Makefile` to `newdir/Makefile`. Finally, change the `SRCS` declaration in `newdir/Makefile` to contain only the source files in that directory and change the line

```
ARCHIVE = libutil$(ENV).a
```

to

```
ARCHIVE = libnewdir$(ENV).a
```

An Example

Given the following Cminus program (3.add.cm):

```
int main () {
int i,j,k,l;

    write(10+20);
    i=1;  k=3; l=4;
    j = i + k + l;
    write(j);
}
```

your expected output in 3.add.s should be

```
30
8
```

Additional Notes (Please read!)

The lexeme of `STRING` token as it is includes the leading and ending double quotes. The actual content of the string should not include the quotes.

In both CminusParser.y and CminusScanner.l, I have renamed the prefix `yy` to `Cminus_`. You will need to use `Cminus_` instead of `yy` as the prefix for the common Bison and Flex functions and global variables. For example, instead of using `yylval`, use `Cminus_lval`.

The `write` statement should output a newline by default.