COSC 362
Project I

Objectives

1. To become familiar with flex, a popular program used to generate lexical analyzers.
2. To implement a lexical analyzer for programs written in TINY.
3. To understand which strings constitute valid tokens in TINY.
4. To practice writing regular expressions by specifying valid TINY tokens.

Machine Details

Use any Linux or MacOS machine. You may need run the command if you use Linux:
sudo apt install flex bison

Assignment Description

This assignment asks you to implement the lexical-analysis phase of a compiler. You will use flex (the current version of lex) to generate a lexical analyzer for programs written in TINY.

Please begin by downloading the auxiliary file *tiny.y in MyClasses.* In *tiny.y* you will find an enumeration of all possible TINY tokens.

Then, in a new file called *tiny.l* (that is the letter "el", which stands for "lex", after the period), write a flex input file that recognizes the token types enumerated in *tiny.y*. Whenever your lexer recognizes a TINY token, it should print the token to the screen (example executions are shown below).

Big Hint

You will probably find it helpful to study the *dism.l* file and use *dism.l* as a reference for your own *tiny.l* file. Your *tiny.l* will probably be quite similar to *dism.l*, except that *tiny.l* will have the DEBUG flag set to 1 instead of 0, and *tiny.l* will have different tokens and regular expressions to match those tokens.

Compilation of the Lexer

Compile your lexer with the following commands (where ">" is a command prompt).

```
> flex tiny.l
> bison tiny.y
> gcc tiny.tab.c –o tiny-lex
```

Example Executions

The *sample.tny* program is:

```
{ Sample program
  in TINY language -
  computes factorial
}
read x; { input an integer }
if 0 < x then { don't compute if x <= 0 }
  fact := 1;
  repeat
    fact := fact * x;
    x := x - 1
  until x = 0;
  write fact  { output factorial of x }
end
```

This *sample.tny* file is correctly tokenized as follows.

```
>./tiny-lex sample.tny

READ ID(x) SEMI
IF NUM(0) LT ID(x) THEN
ID(fact) ASSIGN NUM(1) SEMI
REPEAT
ID(fact) ASSIGN ID(fact) TIMES ID(x) SEMI
ID(x) ASSIGN ID(x) MINUS NUM(1)
UNTIL ID(x) EQ NUM(0) SEMI
WRITE ID(fact)
END ENDOFFILE
```

Formatting Issues

If you ever save your *tiny.l* file on a Windows machine, be sure to run the command
dos2unix tiny.l
on a Linux machine before using flex. This command will remove extra whitespace symbols (e.g., ^M) inserted by Windows, which may affect flex.

Submission Notes

- Type the following pledge as an initial comment in your *tiny.l* file: “I pledge my Honor that I have not cheated, and will not cheat, on this assignment.” Type your name after the pledge. Not including this pledge will lower your grade 50%.
- Upload and submit your *tiny.l* file in MyClasses.