Homework 5 – CMSC 331 – Spring 2019
Due Date April 19, 11:59 PM

The goal of this exercise is to acquaint yourself with two very powerful tools for the generation of compilers: *flex* and *bison*. In this homework files for a calculator application are provided to you. The files are as the following list.

File	**Description**
calc.l	Contains the regular expression rules
calc.y	Contains the grammar rules
calc.h	Contains a data structure used by calculator
mymake	Contains the make commands to build the application

You need to transfer the files to a directory in your gl account. The address of gl account is gl.umbc.edu. You need to use a FTP application to transfer the files.

The following command builds the calculator application:

```
[user@linux2] make -f mymake calc
```

After building the application you can run the executable file *calc* which is created in the folder. The following example presents a session of running the calculator. At the prompt of calculator user can enter any arithmetic expression and by pressing enter the calculator evaluates the arithmetic expression. The exit command is dot, and by typing ? character user can get help instructions on the calculator prompt.

```
[user@linux2] calc
>> 3 + 4
7
>> .
[user@linux2]
```

Your job is to modify the application to add logical operators as in the following table. All operators will have left associativity.

Operator	**Meaning**	**Precedence**	**Description**
&	Logical and	Lowest	This is equivalent to && in C language. It is a binary operator. If it evaluates to true, it returns 1 and if evaluates to false it returns 0.
|	Logical or	Same as &	This is equivalent to || in C language. It is a binary operator. If it evaluates to true, it returns 1 and if evaluates to false it returns 0.
==	Equal	Higher than & and |, lower than >	It is a binary operator. If both sides are of equal values, it returns 1, otherwise it returns 0.
>	Greater than	Highest	It is a binary operator. If left operand has a greater value than right operand, it returns 1, otherwise it returns 0.

The following is an example test session that your application can be tested against. You need to test your modifications for all four operators you have added.

```
>> a = 5
>> b = 8
>> a > b
0
```

You need to submit your modified files, i.e. *calc.l* and *calc.y* through the submit utility on your gl account.

Hints:

- In this application you do not need to modify the grammar rules to implement associativity and precedence of operators. They can be defined for *bison*, and *bison* will implement them.
- In the declaration section of *calc.y* you can define the precedence and associativity of operators. The precedence and associativity of arithmetic operators are already defined in this file. You can define this information for your logical operators in the same place and in the same way.
- Logical and (&) has a higher precedence than assignment (=) operator, and Greater than (>) has a lower precedence than + and - operators