CSCI 2271 Computer Systems
Assignment 4: Strings
Due Friday, February 19

Your task is to write a program, called *calc.c*, which evaluates an arithmetic expression given on the command line. For example, here is a screenshot of my program in action:

```
adminuser@adminuser-VirtualBox ~/HW/HW4 $ calc 123 + [ 43 x 100 ]
4423
adminuser@adminuser-VirtualBox ~/HW/HW4 $ calc [ 1 + 2 ] x [ 3 + 4 ]
21
```

Note that I use square brackets for parentheses and "x" to denote multiplication. This is because the parenthesis and asterisk characters have special meanings when they appear on the command line.

Let's use the word "token" to denote an operator, parenthesis, or number in the expression. Assume that these tokens are separated by spaces on the command line. Then the *argv* array will contain one element for each token.

You probably learned how to do expression evaluation in your data structures class. The idea is to use two stacks: a stack of strings (called the "operand stack") and a stack of numbers (called the "value stack"). Use the following algorithm:

```
For each token t:
   If t is an operator or a left parenthesis, push it on the operand stack.
   Else if t is a right parenthesis:
     Repeatedly pop the operand stack until you hit a left parenthesis.
     For each operator found:
        Pop the value stack twice to get the operands.
        Perform the operation on these operands.
        Push the result of the operation on the value stack.
   Else t denotes a number.
     Convert t to an integer, and push it on the value stack.

When you are out of tokens, repeatedly pop the operand stack until it is
empty, performing each operation as before. At this point there should be
one number on the value stack, which is the answer.
```

Your program only needs to implement two operators: "+" for addition, and "x" for multiplication. These operations both take two operands.

You should implement each stack as an array. The value stack is simply an array of ints. The operand stack should be an array of *char* * pointers – that is, each element of the array denotes a string. You can assume that each stack will never grow larger than

50 elements. You should associate an integer with each stack that keeps track of the top of the stack.

Your code must implement the following functions:

```
void pushValStack(int stack[],  int *top, int value);
int  popValStack (int stack[],  int *top);
void pushOpStack(char *stack[], int *top, char *value);
char *popOpStack(char *stack[], int *top);
```

Use these functions when you need to push and pop one of the stacks. You are of course free to write additional functions as well.

There are numerous ways that your program could blow up – for example, your input expression could have unbalanced parentheses. You are not required to check for bad input.

When you are finished, submit your *calc.c* file to Canvas.