CS 201, Spring 2020
Homework Assignment 4

Due: 23:55, May 18, 2020

In this homework assignment, you are supposed to implement a program for manipulating algebraic expressions of infix and prefix forms. Your implementation should enable the user to convert an expression of infix form to prefix form as well as to evaluate prefix expressions.

Your implementation MUST include the following global functions whose details are given below. Note that we will use these function prototypes to test your programs. Thus, you are not allowed to change the prototypes of these functions. On the other hand, you are free to add as many additional functions as you would like into your implementation.

```
// It converts an infix expression exp to its equivalent prefix form.
string infix2prefix( const string exp );

// It evaluates a prefix expression.
double evaluatePrefix( const string exp );
```

You MUST use stacks in your implementation. Thus, you will define a class for the ADT stack and use it in your functions. Note that you are allowed to use the C++ codes that are given in your textbook.

In your algorithms, you may have the following assumptions:

1. The input strings are syntactically correct expressions. For example, you may assume that the string is a valid infix expression when the `infix2prefix` function is called or you may assume that the string is a valid prefix expression when the `evaluatePrefix` function is called.

2. An input string can include only digits and binary operators. You may assume that only the binary operators *, /, +, and - are allowed (no unary and exponentiation operators are allowed). Moreover, you may assume that operands are single digits (digits from 1 to 9). In case of infix expressions, the string can also include parentheses in addition to digits and binary operators. For simplicity, you may assume that there are no spaces between the digits and the operators.

Here is an example test program that uses these functions. We will use a similar (but different) program to test your solution so make sure that you test your solutions thoroughly.

```
#include "AlgebraicExpression.h"

int main() {

    cout << infix2prefix("(4+5)-2*4") << endl;
    cout << evaluatePrefix("/4+3*-528") << endl;

    return 0;
}
```

Notes:

1. This assignment is due by 23:55 on Monday, May 18, 2020. You should upload your homework to the Moodle page of the course. This upload link will be available between May 10 and May 23. No hardcopy submission is needed. The standard rules about late homework submissions apply. Please see the updated course syllabus for further discussion of the late homework policy as well as academic integrity.

2. In this assignment, you must have separate interface and implementation files (i.e., separate `.h` and `.cpp` files). We will test your implementation by writing our own driver `.cpp` file which will include your header files. For this reason, your files' names MUST BE "`Stack.h`", "`Stack.cpp`", "`AlgebraicExpression.h`" and "`AlgebraicExpression.cpp`". You should upload these files (and any additional files you may have in your solution) as a single zip archive file. The submissions that do not obey these rules will not be graded. You must also write your own driver file, which must be different from the one that we gave above, to test each of your functions. You MUST submit this test code as well in the archive file.

3. You **MUST** use the ADT stack in your implementation. Thus, you will define and implement a class for the ADT stack and use it in your functions. Note that you are allowed to use the C++ codes that are given in your textbook but are not allowed to use any other stack implementation. For example, you cannot use the stack class defined in another book or on a web site. Similarly, you are not allowed to use any functions in the C++ standard template library (STL).

4. Your code must not have any memory leaks. You will lose points if you have memory leaks in your program even though the outputs of the operations are correct.

5. You are free to write your programs in any environment (you may use either Linux or Windows). On the other hand, we will test your programs on "`dijkstra.ug.bcc.bilkent.edu.tr`" and we will expect your programs to compile and run on the "`dijkstra`" machine. If we could not get your program properly work on the "`dijkstra`" machine, you would lose a considerable amount of points. Therefore, we recommend you to make sure that your program compiles and properly works on "`dijkstra.ug.bcc.bilkent.edu.tr`" before submitting your assignment.

6. This homework will be graded by your TA Aydamir Mrzayev (aydamir.mirzayev@bilkent.edu.tr). Thus, you may ask your homework related questions directly to him.