CS 201, Summer 2022
Homework Assignment 4

Due: 23:55, July 27, 2022

Important Notes:

Please do not start the assignment before reading these notes.

1. This assignment is due by 23:55, July 27 2022. You should upload your homework to the upload link on Moodle before the deadline. Please see the course web page for the policy on late sumissions and 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 if you wrote additional classes in your solution) as a single archive file (.zip file). We also recommend you to write your own driver file to test each of your functions. However, you MUST NOT submit this test code (we will use our own test code). In other words, your submitted code should not include any `main` function.
3. You **MUST** use 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 allow to use any other list implementation (e.g., you cannot use the list class defined in another book). Similarly, you are not allow 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 Mert Kara (mert.kara at bilkent.edu.tr). Thus, you may ask your homework related questions directly to him.

In this homework assignment, you will implement a program for manipulating algebraic expressions of infix and postfix forms. Your implementation should enable us to convert an expression of infix form to postfix form as well as to evaluate postfix 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.

```cpp
// It converts an infix expression exp to its equivalent postfix form.
string infix2postfix( const string exp );

// It evaluates a postfix expression.
double evaluatePostfix( 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 if `infix2postfix` function is called or you may assume that the string is a valid postfix expression if `evaluatePostfix` 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 is no spaces between the digits and 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.

```cpp
#include <iostream>
#include "AlgebraicExpression.h"

int main() {
    cout << infix2postfix("(4+5)-2*4") << endl;
    cout << evaluatePostfix("352-8*+4/") << endl;

    return 0;
}
```