

Project 1

Stacks

Submission Notes

This assignment requires one cpp file submission. Do not upload a code snippet, the only files you submit for the code should be the source files. Do not submit any zip or compressed files, binary files, or any other generated files. Do not put the code inside the PDF file. Submission of unnecessary files or binary files in addition to source files will make you lose the points of the assignment. The code must have the following:

- a. Use consistent indentation through your code.
- b. Use consistent braces style through your code.
- c. File-level comments, these comments will include the file name, your name, course number, and date of last modification.
- d. Function level comments include the function's description, the function's parameters, and the function's return value. These comments will appear before each of the functions, not the prototypes.
- e. In function comments, these comments will include a description of the atomic functionalities within the bodies of the functions.
- f. We evaluate your code using the following C++ online compiler. You will not get points if your assignment does not compile with this.

https://www.onlinegdb.com/online_c++_compiler

Stacks (100 points)

1) string infixToPostfix(string infix) 60 points

Implement a function using stacks to convert an infix expression to its corresponding postfix expression.

- You can use class Stack and stacks functions implemented in the class.
- Assume that the input string (infix expression) has the maximum length of 100.
- Assume that the input string (infix expression) has the following operations only : +, -, /, * (Don't worry about power function.)
- The input infix expression can have parenthesis, (Assume the parenthesis are balanced)
- The infix expression only has single digit operands (0,1,2,3,4,5,6,7,8, and 9).

Example:

input: $3 + 2 * (4 - 1)$

output: $3241 - * +$

- **Hint:** Implement a helper function to assign priority for the operations. For instance, your function, int prio(char c), can get a character and returns an integer for the corresponding character. (Higher integer means higher priority).

2) int evaluatePostfix(string exp) 30 points

Implement a function to evaluate value of a postfix expression.

- You can use the same class Stack and its functions implemented in the previous part.

Example:

input: $3241 - * +$

output: 9

- **Hints:**

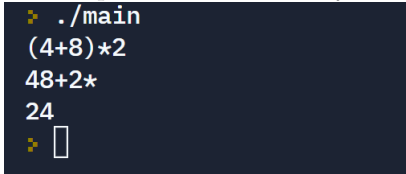
- You can use isdigit(char c) function using `#include <ctype.h>` to check whether a character is a digit or not. For example, isdigit('2') returns TRUE.
- You can use char - '0' to successfully convert a char to int. For example, when you calculate '5'-'0' the compiler replaces that with 53 - 48 which is 5

3) Test your program 10 points

Implement a test program exactly as follows. Once the program is executed, user enters the expression.

```
int main()
{
    string exp;
    cin>>exp;
    string postfix;
    int evaluation;
    postfix = infixToPostfix(exp);
    cout << postfix << "\n";
    evaluation = evaluatePostfix(postfix);
    cout << evaluation << "\n";
    return 0;
}
```

The output should be exactly like the following:



```
./main
(4+8)*2
48+2*
24
█
```

Components of the program:

```
#include <iostream>
#include <string.h>
#include <ctype.h> // needed for isdigit
using namespace std;

#define MAX 100
class Stack {
public:
    int top;
    char exp[MAX]; // Maximum size of Stack
    Stack() {
        top = -1;
    }

    // functions prototypes
    bool push(char item);
    char pop();
    char peek();
    bool isEmpty();
};

// return precedence of operators
int prec(char c);

// convert infix expression to postfix expression
string infixToPostfix(string s);

//evaluate postfix expression
int evaluatePostfix(string exp);
```