CS 211 Data Structures and Algorithms Lab
Spring 2023

Assignment no.	**1**
Objective	Stack Application: Infix to postfix conversion
Total marks	6
Due date (without penalty)	28 April (Friday) 11:59 pm
Penalty for violating naming convention(s)	10%

Task: Implement the infix expression to postfix expression conversion ***using stack operations***.

Description: Your program should read an expression from the input file (which is passed as the command line argument) and has to do the following:

a. Your program should validate whether the scanned expression is an infix expression? as below:
   i. The stack size should be calculated based on the number of operators present in the given input expression.
   ii. If the scanned expression is like "*(a+-b*c)*" or like "*a+b*c/(dy-e)^g^h*" then your program should print output in *output.txt* as "*Invalid expression*".

      Note that the above expression contains two operators one after another, which is an invalid infix expression. Similarly, two operands appeared one after another without an operator in between, which is also an invalid expression.

   iii. If the scanned expression is like "*a*(b+c-d/e^f*g*" or like "*a+b*c)*" then your program should print output in *output.txt* as "*Unequal Parentheses*"

b. If the scanned expression is a valid infix expression then your program has to convert it to a valid postfix expression and print each stack operation and the corresponding postfix expression in the *output.txt*. Sample *output.txt* is given below:

   Example of Infix to postfix:

   Input: *a+b*c/(d-e)^g^h*

   Output: *abc*de-g^h^/+*

Sample of Output.txt:

The output of your program should be in a file named '***output.txt***'. Each line should denote a stack operation. And at last your program should print the result / postfix expression.

Sample output is as below:

```
+ is pushed into stack
* is pushed into stack
* is popped from stack
/ is pushed into stack
( is pushed into stack
- is pushed into stack
- is popped from stack
( is popped from stack
^ is pushed into stack
^ is popped from stack
^ is pushed into stack
^ is popped from stack
/ is popped from stack
+ is popped from stack
Postfix expression is:abc*de-g^h^/+
```

Note:

Limit your above task only for arithmetic operators (+ , - , / , * , % , ^). Precedence chart of these operators is given below:

Precedence	**Operator**	**Description**
4 (Highest)	()	Parentheses
3	^	Bitwise Exclusive OR
2	*, /, %	Multiplication, division, modulus
1 (Lowest)	+, -	Addition, subtraction

How to execute?

Once you have completed writing the code do the following:

- Save it with <roll_no> *.c* extension in a working directory.
- Open the terminal and change the directory to your working directory, and run the following command to compile your program:

```
$gcc <roll_no> .c
```

- If your program is error free (on successful compilation), execute the following command:

  ```
  $ ./a.out input_6.txt
  ```

  Where *input_6.txt* is the input file

Submission and evaluation:

- The program you submit should output: ***output.txt*** when we run the program for evaluation.
- The main file of your program should be named after your IIT Dharwad roll number. Like ***<roll_no>.c***, where ***roll_no*** specifies your IIT Dharwad roll number.Ex: 220010001.c.
- Do the stress test of your program well before submission.
  - You may use the attached sample input files for testing; the corresponding output files are also attached;
  - We have some hidden inputs with us to test your program. *The mark you obtain is purely based on whether your program correctly gives outputs for the hidden inputs*.
- If your program has only a single source file, please submit the file as it is. If your program has multiple source files, please submit your code as a zip file where the name of the zip file should be your roll number. It **is important that you follow the input/output conventions exactly** (including the naming scheme) as we may be doing an automated evaluation. ***There will be a penalty of 10% (on the mark you deserve otherwise) if you do not follow the naming conventions exactly.***
- Follow some coding style uniformly. Provide proper comments in your code.
- *Submit only through moodle*. ***Submit well in advance***. Any hiccups in the moodle at the last minute is never acceptable as an excuse for late submission. Submissions through email or any other means will be ignored.
- Acknowledge the people (other than the instructor and TA) who helped you to solve this assignment. The details of the help you received and the names of the people who helped you (including internet sources, if applicable) should come in the beginning of the main file as a comment. ***Copying others' programs and allowing others to copy your program are serious offenses and a deserving penalty will be imposed if found.***

- To be considered for the evaluation without penalty, you have to submit your program by the due date. ***No single minute relaxation on late submission.***
- If the output is not matching with the expected output, then evaluation is done based on the logic of the code and partial marks will be awarded.