National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B.Tech. (CSE)
CS2092D Programming Laboratory

Assignment 7

Submission deadline (on or before):

• **11:00 PM, Wednesday, 25.10.2023**

Policies for Submission and Evaluation:

• You must submit your assignment in the Eduserver course page, on or before the submission deadline.
• Ensure that your programs will compile and execute without errors in the Linux platform.
• During the evaluation, failure to execute programs without compilation errors may lead to zero marks for that evaluation.
• Detection of ANY malpractice related to the lab course can lead to awarding an '**F**' grade in the course.
• Any queries or clarifications on the assignment questions should be submitted on or before **11:00 PM, Wednesday, 25/10/202**3 in the below spreadsheet:

https://docs.google.com/document/d/1hVCoBUUeaz9PMEDPHPYwlWiD1kmyel1uaCFY_hjs7Uk/edit?usp=sharing
Any queries after that will not be entertained.

Naming Conventions for Submission

• Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar, .tar, .gz).
The name of this file must be

ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>.zip

(Example: ASSG7_BxxyyyyCS_LAXMAN.zip). DO NOT add any other files (like temporary files, input files, etc.) except your source code, into the zip archive.

• The source codes must be named as

ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c

(For example: ASSG7_BxxyyyyCS_LAXMAN_1.c). If you do not conform to the above naming conventions, your submission might not be recognized by our automated tools, and hence will lead to a score of **0**(**ZERO**) marks for the submission. So, make sure that you follow the naming conventions.

Standard of Conduct

• Violation of academic integrity will be severely penalized. Each student is expected to adhere to high standards of ethical conduct, especially those related to cheating and plagiarism. Any submitted work MUST BE an individual effort. Any academic dishonesty will result in zero marks in the corresponding exam or evaluation and will be reported to the department council for record keeping and for permission to assign F grade in the course. The department policy on academic integrity can be found at:
https://minerva.nitc.ac.in/cse/sites/default/files/attachments/news/Academic-Integrity.pdf.

General Instructions

• Programs should be written in C language and compiled using C compiler in Linux platform. Submit the solutions to questions 1 to 4 through the submission link in Eduserver.

Questions

Q1. Jack used C programming to create a calculator. He put the calculator into practice to assess the postfix phrases. A friend of Jack's named Jane has access to the calculator that Jack set up. Jane is interested in it and wants to check out the calculator Jack put in place. But Jane's unfamiliarity with postfix expressions is the problem. Can you assist Jane with the infix to postfix expression conversion so she can use Jack's calculator to evaluate the expression?

Input format:

- First line containing the number of charecters in infix expression.
- Second line contains infix expression seperated with single space in uppercase letters.

Output Format:

- Single line which contains the postfix expression.

Sample input:

7
A + B * C / D

Sample Output:

A B C * D /+

Q2. Imagine you are working on developing a scientific calculator application, and one of the features you need to implement is the conversion of postfix expressions to infix expressions to enhance user readability. As part of your task, you decide to use an inorder traversal to obtain the corresponding infix expression.

Your calculator application supports various mathematical operations such as addition, subtraction, multiplication, division. You have already implemented a postfix expression parser that can handle these operations. Now, you need to convert the postfix expression to infix notation.

Input format:

- A single string, representing a valid postfix expression.The allowed operators are '+', '-', '*', and '/'.

Output Format:

- A string representing the corresponding infix expression.

Sample Input 1:

23+5*

Sample Output 1:

(2+3)*5

Sample Input 2:

42+3*1-

Sample Output 2:

((4+2)*3)-1

Q3. You are developing a simple calculator application for a futuristic handheld device. The calculator uses a unique postfix notation where each operator follows its operands, and the user enters expressions without the need for parentheses. The user can input numbers, basic arithmetic operators (+, -, *, /), and they expect an immediate result.

Implement an application that needs to handle these types of expressions and provide an answer instantly. The application produced by you implements a function that takes user's input in postfix notation and returns the simplified result otherwise it returns '**-1'** if the post fix evaluation is incorrect.

Input format:

- A single string which represents a postfix expression. The characters in string can be [0-9,+,-,*, /]. All characters are separated by one space each.

Output format:

- A single integer, found after evaluating the postfix expression.

Sample Input 1:

2 3 1 * + 9 -

Sample Output 1:

-4

Sample Input 2:

100 200 + 2 / 5 * 7 +

Sample Output 2:

757

Q4. You are a software engineer working on an advanced project to create a powerful tool for manipulating and analyzing mathematical expressions. This tool is essential for engineers and scientists dealing with complicated mathematical computations.

Your program should provide the following features:

- **Infix to Postfix Conversion:** Implement a function that converts an infix mathematical expression to postfix notation.
- **Expression Evaluation:** Implement a function to evaluate a given postfix mathematical expression and display the result. It can handle basic arithmetic operations, including addition, subtraction, multiplication, and division.
- **Expression Tree Analysis:** Implement functions to display the height of the expression tree and find the minimum length from the root to any leaf.

Input Format:

- A single line contains the character 'i', followed by a valid infix expression. This expression is a combination of digits (operands) in the range [0,1000], binary operators '+', '−', '∗', and '/', and special characters '(' and ')'; each separated by a single space.

Output Format:

- The first line display the postorder traversal of the given infix expression as a space separated format.
- The second line is a Real number (Keep only two if there are more than two digits after the decimal point), the result of given expression (Note: If the expression results in an answer divisible by zero, print 'N').
- The third line is an integer, the height of the expression tree.
- The fourth line is an integer, the minimum length from the root to leaf.

Sample Input 1:

((11 / 92) * (33 - 45) + 57 * 74) * (85 / 5)

Sample Output 1:

11 92 / 33 45 - * 57 74 * + 85 5 / *
71681.60
4
2

Sample Input 2:

(2 - 3) * (4 / 0)

Sample Output 2:

2 3 - 4 0 / *
N
2
2