National Institute of Technology, Calicut
Department of Computer Science and Engineering
CS2094 – Data Structures Lab
Assignment 4 (Advanced)

Submission deadline (on or before):
1st March 2016, 10:00:00 PM (for Advanced batch)

Naming Conventions for submission:
Submit a single ZIP (.zip) file (do not submit in any other archived formats like .rar or .tar.gz). The name of this file must be ASSG<Number>A_<ROLLNO>_<FIRSTNAME>.zip
(For example: ASSG4A_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>A_<ROLLNO>_<FIRST NAME>_<PROGRAM-NUMBER>.<extension> (For example: ASSG4A_BxxyyyyCS_ LAXMAN_1.c)

Checking your assignment with sample test cases:
You must check your programs in this assignment with the sample input test case files. Also, verify your output with the sample output files. Your programs will be evaluated using the script which compiles all your programs and verify your output with the correct output. Hence, do not print any extra messages while reading the input from the user as well as printing the output.

Questions:

1. Given the preorder and inorder traversal of a binary tree with unique keys, design and implement an algorithm to create the binary tree and print it in a scheme like format.

Input Format:
- The first line contains a positive integer, representing the number of nodes in the tree.
- The second line contains n space separated integers, representing a valid preorder traversal of a tree.
- The third line contains n space separated integers, representing a valid inorder traversal of the tree.

Output format:
Single line representing the binary tree in a scheme like format.

Example 1:

Sample Input:
```
10
43 15 8 30 20 35 60 50 82 70
8 15 20 30 35 43 50 60 70 82
```
Sample Output:
```
( 43 ( 15 ( 8 ( ) ( ) ) ( 30 ( 20 ( ) ( ) ) ( 35 ( ) ( ) ) ) ) ( 60 ( 50 ( ) ( ) ) ( 82 ( 70 ( ) ( ) ) ( ) ) ) )
```

Example 2:

Sample Input:
```
11
55 28 74 96 20 59 42 13 52 82 44
96 74 28 59 20 42 55 13 82 52 44
```
Sample Output:
```
( 55 ( 28 ( 74 ( 96 ( ) ( ) ) ( ) ) ( 20 ( 59 ( ) ( ) ) ( 42 ( ) ( ) ) ) ) ( 13 ( ) ( 52 ( 82 ( ) ( ) ) ( 44 ( ) ( ) ) ) ) )
```

2. Given a valid postfix expression, create the corresponding expression tree (each node of the tree may have a pointer to its parent, in addition to the left and right pointers). Traverse this tree iteratively in inorder, preorder and postorder fashion without using a stack. The program must support (at least) the following in the postfix expression:

Binary operators:
^ : Exponentiation (Highest precedence)
/, * : Division, Multiplication
+, - : Addition, Subtraction(Lowest precedence)
Operands:
The operands are all variables, which is represented by a single lowercase English character (a-z)

Input Format

- The only line of the input contains a valid postfix expression. (There will not be any space anywhere in the expression)

Output Format

- The inorder, preorder and postorder traversals. The results of each traversal should be on a fresh line.
- For the inorder traversals, use proper parentheses also (i.e., each operator, together with its operands must be enclosed in parentheses).

Example 1:

Sample Input:
abc*+

Sample Output:
Inorder: (a+(b*c))
Preorder: +a*bc
Postorder: abc*+

Example 2:

Sample Input:
m
Sample Output:
Inorder: m
Preorder: m
Postorder: m

Example 3:

Sample Input:
abcd^*+e-
Sample Output:
Inorder: ((a+(b*(c^d)))-e)
Preorder: -+a*b^cde
Postorder: abcd^*+e-

3. Write an iterative program to do a zigzag level order traversal of a binary tree. A zigzag level order traversal starts with the root node, then visits nodes in the first level from right to left, then nodes in the second level from left to right, then nodes in the third level from right to left, and so on.

Input Format:

- Single line representing the binary tree in a scheme like format.

Output format:

- *n* space separated integers, representing the zigzag level order traversal of the tree.

Example 1

Sample Input:
(43 (15 (8 ( ) ( )) (30 (20 ( ) ( )) (35 ( ) ( )))) (60 (50 ( ) ( )) (82 (70 ( ) ( )) ( ))))
Sample Output:
43 60 15 8 30 50 82 70 35 20

Example 2

Sample Input:

(1 (2 (4 (7 ( ) ( )) (8 ( ) ( ))) (5 (9 ( ) ( )) ( ))) (3 ( ) (6 (10 ( ) ( )) (11 ( ) ( )))))

Sample Output:

1 3 2 4 5 6 11 10 9 8 7

****