National Institute of Technology, Calicut

Department of Computer Science and Engineering

CS2094D – Data Structures Lab

Assignment-1 (Advanced Batch)

Policies for Submission and Evaluation

You must submit your assignment in the moodle (Eduserver) course page, on or before the submission deadline. Also, ensure that your programs in the assignment must compile and execute without errors in Athena server. During evaluation your uploaded programs will be checked in Athena server only. Failure to execute programs in the assignment without compilation errors may lead to zero marks for that program.

Your submission will also be tested for plagiarism, by automated tools. In case your code fails to pass the test, you will be straightaway awarded zero marks for this assignment and considered by the examiner for awarding F grade in the course. Detection of ANY malpractice regarding the lab course will also lead to awarding an F grade.

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>_<ROLLNO>_<FIRSTNAME>.zip (For example: ASSG4_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>_<FIRSTNAME>_<PROGRAM-NUMBER>.<extension>
(For example: ASSG4_BxxyyyyCS_LAXMAN_1.c). If there is a part a and a part b or a particular question, then name the source files for each part separately as inASSG4_BxxyyyyCS_LAXMAN_1b.c.

If you do not conform to the above naming conventions, your submission might not be recognized by some automated tools, and hence will lead to a score of 0 for the submission. So, make sure that you follow the naming conventions.Standard of Conduct Violations 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:
http://cse.nitc.ac.in/sites/default/files/Academic-Integrity.pdf .

<u>**Assignment Questions**</u>

1. Write a menu driven program to implement a C program to get the highest occuring element in a Linked List of integer data. It may be recalled that the data associated with the node should not be changed. Only pointers can be changed. Implement the program with best time and space complextity possible. Input should be read from the file **input.txt** and output should be written to the file **output.txt** Your program should include the following functions.

main() - repeatedly reads a character 'c', 'p’, 'h', or 'a' from the terminal and calls the given functions appropriately until character 's' is entered.

create()- read the elements of linked list from the terminal and create linked list.

print()- function to print the linked list.

h_occur()-function to print the highest occurring element in the linked list.

add_node()- function to add the new element in the linked list.

Input Format

a) The input consists of multiple lines. Each line contains a character from {**'c', 'p’, 'h', 'a'** } followed by zero or n integers. Assume 1 <= n <= 20

b) Character **'c'**: character ‘c’ specifies create linked list with ‘n’ nodes. The next line contains ‘n’ integers separated by space, which are the elements of the linked list. Create an initial linked list with the ‘n’ elements.

c) Character **'p'**: prints the linked list.

d) Character **'h':** prints the highest occurring element. (In case of more than One elements print the first element)

e) Character '**a**': character ‘a’ is followed by an integer k. Inserts a new node at the end with value k to the linked list.

f) Character **'s'**: “terminates the program”.

Output format:

The output (if any) of each command should be printed on a separate line

Sample Input:

```
c 3
2 2 8
p
```

```
h
a 8
a 8
h
```

Sample Output:

```
2 2 8
2
8
```

2. Write a program to convert an infix expression into postfix expression.

Infix expression:

The expression of the form **a op b**. When an operator is in-between every pair of operands.

Postfix expression:

The expression of the form **a b op**. When an operator is followed for every pair of operands.

Input should be read from the file **input.txt** and output should be written to the file **output.txt**

Your program should include the following functions.

infixToPostfix(expression) - Converts infix expression to postfix expression.

Print( ) - Displays the final converted expression. .

Input File Format

The input consists of multiple lines of infix expression, each line of the input contains combination of characters(operands) from {a to z} and symbols(operator) like +,-,*,/,(,) and ^

Output File Format:

The output (if any) of each command should be printed on a separate line.

Sample Input:

```
a + b * c
( a + b ) * c
( a + b ) * ( c + d )
```

Sample Output:

```
a b c * +
a b + c *
a b + c d + *
```

3. Write a program for the evaluation of a postfix expression. Input should be read from the file **input.txt** and output should be written to the file **output.txt** Your program should include the following functions.

evaluatePostfix(exp) -evaluates postfix expression.

Print( ) - prints the final output.

Input File Format

The input consists of multiple lines of postfix expression. Each line of the input contains combination of numbers and symbols like +,-,*,/,(,) and ^

Output file format:

The output (if any) of each command should be printed on a separate line

Sample Input:

```
2 3 1 * + 9 -
100 200 + 2 / 5 * 7 +
```

Sample Output:

```
-4
757
```

4. Write a program to create a Binary Search Tree (BST). Input should be read from the file **input.txt** and output should be written to the file **output.txt** Your program should include the following functions.

insert(tree, element) – adds the node specified by element (which contains the data) into the BST specified by tree.

search(tree, key) – searches for the data specified by key in the BST specified by tree.

findMin(tree) – retrieves the smallest data in the BST specified by tree.

findMax(tree) – retrieves the largest data in the BST specified by tree.

predecessor(tree, element) – retrieves the inorder-predecessor of the node specified by element in the BST specified by tree.

successor(tree, element) – retrieves the inorder-successor of the node specified by element in the BST specified by tree.

delete(tree, element) – removes the node specified by element from the BST specified by tree.

inorder(tree) – To do a recursive inorder traversal of the BST.

preorder(tree) – To do a recursive preorder traversal of the BST.

postorder(tree) – To do a recursive postorder traversal of the BST.

Input format

The input consists of multiple lines, each containing a four letter string followed by zero or one integer. The meaning of each line is given below:

String **“stop”** means stop the program.

String “**insr**” means, create a node which contains the next integer(>= 0) from the input as the data part, and then, insert this node into the BST. In this case, the data is given on the same line as the string “**insr**”, separated by a space.

String “**srch**” means, search for the key specified by the next integer(>= 0) from the input, in the BST. In this case, the key is given on the same line as the string “**srch**”, separated by a space. If the search is successful, output “**FOUND**”. Otherwise, output “**NOT FOUND**”.

String “**minm**” means find and output the minimum value in the BST. Print “**NIL**” if the BST is empty.

String “**maxm**” means find and output the maximum value in the BST. Print “**NIL**” if the BST is empty.

String “**pred**” means find and output the inorder-predecessor of the data specified by the next integer(>= 0) from the input. In this case, the data is given on the same line as the string “**pred**”, separated by a space. Output “**NIL**”, if the data exists in the tree, but there is no inorder-predecessor for the data. Output “**NOT FOUND**”, if the data is not present in the tree.

String “**succ**” means find and output the inorder-successor of the data specified by the next integer(>= 0) from the input. In this case, the data is given on the same line as the string “succ”, separated by a space. Output “**NIL**”, if the data exists in the tree, but there is no inorder-successor for the data. Output “**NOT FOUND**”, if the data is not present in the tree.

String “**delt**” means delete the data specified by the next integer(&gt;= 0) in the input from the BST, if present. In this case, the data is given on the same line as the string “**delt**”, separated by a space. (Here, the data to be deleted is guaranteed to be present in the BST).

String “**inor**” means output the in-order traversal of the BST.

String “**prer**” means output the pre-order traversal of the BST.

String “**post**” means output the post-order traversal of the BST.Output format

The output (if any) of each command sh

ould be printed on a separate line.

Note : Strictly follow the output format . It should be NIL, NOT FOUND, FOUND

Sample Input

```
srch 25
minm
maxm
pred 25
succ 25
insr 25
srch 25
minm
maxm
pred 25
succ 25
insr 13
insr 50
insr 45
insr 55
insr 18
srch 10
inor
prer
post
delt 55
delt 13
delt 25
minm
maxm
stop
```

Sample Output

```
NOT FOUND
NIL
NIL
NOT FOUND
NOT FOUND
FOUND
```

```
25
25
NIL
NIL
NOT FOUND
13 18 25 45 50 55
25 13 18 50 45 55
18 13 45 55 50 25
18
50
```

5. Given the preorder traversal with unique keys, construct the binary search tree,

Input format

The first line contains a positive integer, representing the number of nodes in the tree.

The second line contains, in space separated integers, representing a valid preorder traversal of a binary search tree.

Output format

Single line representing the binary tree in the following format :

Tree t is represented as (t (left-subtree) (right-subtree)). Empty parentheses ( ) represents a null tree. A leaf node l is represented as (l)

Sample Input

```
10
43 15 8 30 20 35 60 50 82 70
```

Sample Output

```
( 43 ( 15 ( 8 ( ) ( ) ) ( 30 ( 20 ( ) ( ) ) ( 35 ( ) ( ) ) ) ) ( 60 ( 50 ( ) ( ) ) ( 82 ( 70 ( ) ( ) ) ( ) ) ) )
```

6. Write a recursive program to find height, diameter and maximum width of a binary tree

Input should be read from the file **input.txt** and output should be written to the file **output.txt**

Your program should include the following functions.

height(tree) – Returns the height of a binary tree, which is the number of edges between the tree's root and its furthest leaf. The Height of binary tree with single node is taken as zero.

diameter(tree) – returns the diameter of a tree,T is the largest of the following quantities:

- the diameter of T's left subtree

- the diameter of T's right subtree
- the longest path between leaves that goes through the root of T

maxwidth(tree) – returns maximum Width of a binary tree(width of a level is the number of nodes in that level and maximum width is the maximum among them)

Input format

Single line representing the binary tree in the following format :
Tree t is represented as (t (left-subtree) (right-subtree)). Empty parentheses ( ) represents a null tree. A leaf node l is represented as (l)

Output fomat

Three integers separated by space denoting height, diameter, maximum width respectively

Sample Input

```
( 43 ( 15 ( 8 ( ) ( ) ) ( 30 ( 20 ( ) ( ) ) ( 35 ( ) ( ) ) ) ) ( 60 ( 50 ( ) ( ) ) ( 82 ( 70 ( ) ( ) ) ( ) ) ) )
```

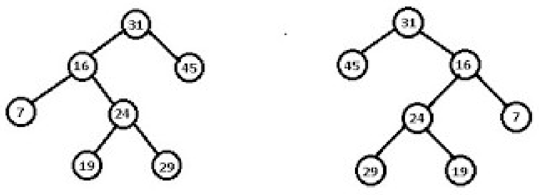
Sample Output

```
3 7 4
```

7. Write a menu driven program to construct a binary tree **t**, using the node format

Data
Left Child
Right Child

and convert it to its mirror tree <u>using recursion</u>. Each node in the binary tree has a key value, which is an integer in the range 2^{-30} to 2^{30}, a pointer to its left child and a pointer to its right child. Refer to the figure below:



Original Tree Mirror Tree

You may declare a global variable ROOT, which points to the root node of the binary tree t.

Your program must implement the following functions:

main ( ) - repeatedly reads a character 'c','p' or 'm' from the terminal and calls the sub functions appropriately until character 's' is entered.

create_tree() - reads the binary tree in parentheses format from the terminal, constructs the binary tree and returns the pointer to the root node of the tree.

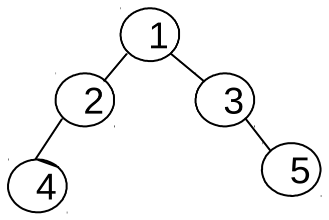
print_tree(t) - prints the binary tree t in the parentheses format. Prints “()” if the binary tree t is empty.

mirror(t) - recursive function to convert the binary tree t to its mirror tree.

Input format:

The input consists of multiple lines, each line starts with a character from {'c', 'p', 'm', 's'}.

Character 'c': creates a binary tree t from its parentheses format*, which is given in the same line as ‘c’and assigns it to the ROOT.



Parentheses notation : (1 (2 (4) ( )) (3 ( ) (5)))

Character 'p' : prints the binary tree pointed to by ROOT, in parentheses format. Print “( )” if the tree is empty.

Character 'm': Convert the binary tree pointed to by ROOT to its mirror tree

Character 's': “terminates the program”.

Output format:

The output (if any) of each command should be printed on a separate line.

Sample Input:

```
c ( 6 ( 9 ( 17 ( ) ( 5 ( 2 ) ( ) ) ) ( 14 ) ) ( 8 ( ) ( 11 ) ) )

p
m
p
s
```

Sample Output:

(6 (9 (17 ( ) (5 (2) ( ))) (14)) (8 ( ) (11)))

(6 (8 (11) ( )) (9 (14) (17 (5 ( ) (2)) ( ))))