National Institute of Technology, Calicut
Department of Computer Science and Engineering
CS2092D – Programming lab
Assignment 4

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 topass 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

Assignment Questions

1. Write a C program to create a Binary Tree and search for an element in the 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.
   - **create(tree)** – creates the Binary Tree specified by ***tree***.
   - **search(tree, key)** – searches for the data specified by ***key*** in the Binary Tree specified by ***tree***.
   - **inorder(tree)** – performs recursive inorder traversal of the Binary Tree specified by ***tree***.

   Input Format:

   The input consists of multiple lines, each containing a letter followed by zero or one integer. The meaning of each line is given below:
   - The first line represents the binary tree in the following format:
     - Tree rooted at ***t*** is represented as (***t*** (***left-subtree***) (***right-subtree***)). Empty parentheses ( ) represents a null tree. A leaf node ***l*** is represented as (***l***).
   - Character **s** means**,** search for the key in the Binary Tree. In this case, the key is given on the same line as the string **s**, separated by a space. If the search is successful, output 'FOUND'. Otherwise, output 'NOT FOUND'.
   - Character **i** means output the in-order traversal of the Binary Tree, each element separated by a space.
   - Character **t** means terminate the program.

   Output Format:

   The output (if any) of each command should be printed on a separate line.
   Note : Strictly follow the output format. It should be NOT FOUND, FOUND.

   Sample Input

   (43 (15 (8 ( ) ( )) (30 (20 ( ) ( )) (35 ( ) ( )))) (60 (50 ( ) ( )) (82 (70 ( ) ( )) ( ))))
   s 15
   s 25
   s 30
   i
   s 55
   t

Sample Output
FOUND
NOT FOUND
FOUND
8 15 20 30 35 43 50 60 70 82
NOT FOUND

2. Write a C program to create a Binary Tree and delete an element from the 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.

- **create(tree)** – creates the Binary Tree specified by ***tree***.
- **delete(tree, element)** – removes the node specified by ***element*** from the Binary Tree specified by ***tree***.
- **inorder(tree)** – performs recursive inorder traversal of the Binary Tree specified by ***tree***.

Input Format:

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

- The first line represents the binary tree in the following format:
    - Tree rooted at ***t*** is represented as (***t*** (***left-subtree***) (***right-subtree***)). Empty parentheses ( ) represents a null tree. A leaf node ***l*** is represented as (***l***).
- Character **d** means delete the data specified by the next integer in the input from the Binary Tree. In this case, the data to be deleted is given on the same line as the string **d**, separated by a space. (Here, the data to be deleted is guaranteed to be present in the Binary Tree).
- Character **i** means output the in-order traversal of the Binary Tree, each element separated by a space.
- Character **t** means terminate the program.

Output Format:
Single line contains **n** space separated integers, representing a valid inorder traversal of the tree.

Sample Input
(43 (15 (8 ( ) ( )) (30 (20 ( ) ( )) (35 ( ) ( )))) (60 (50 ( ) ( )) (82 (70 ( ) ( )) ( ))))
i
d 15
i
d 60

i
t

Sample Output

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

3. 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 the format given in the output format. 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.

- **create(tree)** – create the Binary Tree specified by ***tree***.
- **print(tree)** – print the Binary Tree specified by ***tree*** in the format given in the sample output.

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 the following format
    - Tree rooted at ***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 1

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

Sample Output 1

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

Sample Input 2
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 2
(55 (28 (74 (96 ( ) ( )) ( )) (20 (59 ( ) ( )) (42 ( ) ( )))) (13 ( ) (52 (82 ( ) ( )) (44 ( ) ( ))))

4. Write a program to create a Binary Search Tree (BST) and search for an element in the 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.

- **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***.
- **inorder(tree)** – performs recursive inorder traversal of the BST specified by ***tree***.

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 **"inor"** means output the in-order traversal of the BST.

Output Format:

The output (if any) of each command should be printed on a separate line.
Note : Strictly follow the output format . It should be NOT FOUND, FOUND.

Sample Input
srch 25
insr 25

srch 25
insr 13
insr 50
insr 45
insr 55
insr 18
srch 10
inor
stop

Sample Output
NOT FOUND
FOUND
NOT FOUND
13 18 25 45 50 55

5. Write a program to create a Binary Search Tree (BST) and find the minimum and maximum elements in the 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.

- **insert(tree, element)** – adds the node specified by ***element*** (which contains the data) into 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***.
- **inorder(tree)** – performs recursive inorder traversal of the BST specified by ***tree***.

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 **"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 **"inor"** means output the in-order traversal of the BST.

Output Format:

The output (if any) of each command should be printed on a separate line.
Note : Strictly follow the output format . It should be NIL, NOT FOUND, FOUND.

Sample Input

```
minm
maxm
insr 25
minm
maxm
insr 13
insr 50
insr 45
insr 55
insr 18
inor
minm
maxm
stop
```

Sample Output

```
NIL
NIL
25
25
13 18 25 45 50 55
13
55
```

6. Write a program to create a Binary Search Tree (BST) and find the inorder predecessor and successor of a given element in the 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.

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

- **inorder(tree)** – performs recursive inorder traversal of the BST specified by ***tree***.

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 **"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 **"inor"** means output the in-order traversal of the BST.

Output Format:

The output (if any) of each command should be printed on a separate line.
Note : Strictly follow the output format . It should be NIL, NOT FOUND, FOUND.

Sample Input

```
pred 25
succ 25
insr 25
pred 25
succ 25
insr 13
insr 50
insr 45
insr 55
insr 18
inor
```

pred 45
succ 45
stop

Sample Output
NOT FOUND
NOT FOUND
NIL
NIL
NOT FOUND
13 18 25 45 50 55
25
50

7. Write a program to create a Binary Search Tree (BST) and delete an element from the 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.

- **insert(tree, element)** – adds the node specified by ***element*** (which contains the data) into the BST specified by ***tree***.
- **delete(tree, element)** – removes the node specified by ***element*** from the BST specified by ***tree***.
- **inorder(tree)** – performs recursive inorder traversal of the BST specified by ***tree***.

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 **"delt"** means delete the data specified by the next integer(>= 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.

Output Format:

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

Sample Input

insr 25
insr 13
insr 50
insr 45
insr 55
insr 18
inor
delt 55
delt 13
delt 25
inor
stop

Sample Output

13 18 25 45 50 55
18 45 50

8. Write a program to create a Binary Search Tree (BST) and perform different binary tree traversals. 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***.
- **inorder(tree)** – performs recursive inorder traversal of the BST specified by ***tree***.
- **preorder(tree)** – performs recursive preorder traversal of the BST specified by ***tree***.
- **postorder(tree)** – performs recursive postorder traversal of the BST specified by ***tree***.

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 **"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 should be printed on a separate line.

Sample Input
insr 25
insr 13
insr 50
insr 45
insr 55
insr 18
inor
prer
post
stop

Sample Output
13 18 25 45 50 55
25 13 18 50 45 55
18 13 45 55 50 25

9. Given the preorder traversal of a binary search tree with unique keys, construct the binary search 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.

- **create(tree)** – create the Binary Search Tree.
- **print(tree)** – print the Binary Search Tree in the format given in the sample output.

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 rooted at ***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 ( ) ( )) ( ))))

10. 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(T)** – 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(T)** – returns the diameter of a tree ***T***, which is defined as 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(T)** – returns the 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 rooted at ***t*** is represented as (***t*** (***left-subtree***) (***right-subtree***)). Empty parentheses ( ) represents a null tree. A leaf node ***l*** is represented as (***l***).

Output Format:

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