Department of Computer Science and Engineering

National Institute of Technology Calicut
CS2094D DATA STRUCTURES LABORATORY
(Second Semester MCA)

ASSIGNMENT 1

General Instructions

1. **Try to complete all the assignment questions and submit in eduserver platform**
2. **Your assignment is due by 10am on 24/03/2022**
3. **Implementation and submission of the first two questions are mandatory**
4. **All programs should be implemented in C language**
5. **You have to upload the complete code as a zip file named as Firstname-Regno-Assg#(e.g., John-m200034ca-Assg1. Zip)**
6. **Students who submit works that are found to have been plagiarized can be cause for an automatic 0 marks**
7. **All submitted code should be well-documented, and the code should be easily understood by reading the comments**

Questions

1. The **Binary Search Tree (BST)** data structure supports many of the dynamic-set operations. A BST is organized as a binary tree in which each node is an object that contains a *key* value. In addition to a key and satellite data, each node contains attributes *left*, *right*, and *p* that point to the nodes corresponding to its left child, its right child, and its parent, respectively. If a child or the parent is missing, the appropriate attribute contains the value NIL. The root node is the only node in the tree whose parent is NIL. The keys in a binary search tree are always stored in such a way as to satisfy the ***binary-search-tree*** property:

- Let x be a node in a binary search tree. If y is a node in the left subtree of x, then $y.key < x.key$. If y is a node in the right subtree of x, then $y.key \geq x.key$.

Write a program to create a Binary Search Tree **T** and perform the operations insertion, deletion, search, find minimum, and maximum on **T**.

Input should be read from the console and output should be shown in console. Your program should include the following functions.

- ***main()*** - creates the Binary Search Tree **T** with **T** as the root node (which is NIL initially) and repeatedly reads a character ‘a’, ‘d’, ‘s’, ‘m’, ‘x’, or ‘e’ from the console and calls the sub-functions appropriately until character ‘e’ is entered.
- ***createNode(k)*** - creates a new node with key value *k* and returns a pointer to the new node. All the pointer attributes of the new node are set to NIL.
- ***insert(T, x)*** - inserts the node x into the BST T.
  Note: The caller of this function is assumed to create the node x using the CreateNode() function.
- ***delete(T, x)*** - deletes the node x from the BST T.
  Note: The caller of this function is assumed to invoke Search() function to locate the node x.
- ***search(T, k)*** - searches for a node with key k in T, and returns a pointer to a node with key k if one exists; otherwise, it returns NIL.
- ***minValue(T)*** - returns the minimum value in the BST T.
- ***maxValue(T)*** - returns the maximum value in the BST T.

Input format:

- Each line contains a character from ‘a’, ‘d’, ‘s’, ‘m’, ‘x’, or ‘e’ followed by at most one integer.
- Character ‘a’ is followed by an integer separated by space. In this operation, a node with this integer as key is created and inserted into T.
- Character ‘d’ is followed by an integer separated by space. In this operation, the node with this integer as key is deleted from T and the deleted node’s key is printed.
- Character ‘s’ is followed by an integer separated by space. This operation is to find the node with this integer as key in T.
- Character ‘m’ is to find the minimum value of T.
- Character ‘x’ is to find the maximum value of T.
- Character ‘e’ is to ‘exit’ from the program.

Output Format:

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

- For option ‘d’, print the deleted node’s key. If a node with the input key is not present in T, then print -1.
- For option ‘s’, if the key is present in T, then print 1. If the key is not present in T, then print -1.
- For option ‘m’, print the minimum value of T.
- For option ‘x’, print the maximum value of T.

Sample Input	**Output**
a 25	

a 13	
a 50	
a 45	
a 55	
a 18	
m	13
x	55
s 10	-1
s 25	1
d 13	13
d 10	-1
d 25	25
s 25	-1
e	

(2 marks)

2. A Binary Tree is a rooted tree in which each node is an object that contains a *key* value. In addition to key and satellite data, each node contains attributes *left*, *right*, and *p* that point to the nodes corresponding to its left child, its right child, and its parent, respectively. If a child or the parent is missing, the appropriate attribute contains the value NIL. The root node is the only node in the tree whose parent is NIL. The **Parenthesis Representation** of a binary tree is recursively defined as given below.

- The string ( ) represents an empty tree.
- The string (k left-subtree right-subtree) represents a tree whose root node has key k, left subtree is the left subtree of the root node in **Parenthesis Representation** and right subtree is the right subtree of the root node in **Parenthesis Representation**.

Write a program to create a binary tree T and print T in **Parenthesis Representation**. Your program should contain the following functions:

- ***insert(T, k)*** - inserts the element k to the tree T.
  Note: To Insert an element k into the tree, traverse the tree level by level from left to right, find the first empty child and insert the new key k there, ie the tree remains as a complete binary tree.
- ***print(T)*** - that should take as input a pointer to the root node of a binary tree and print the tree in its **Parenthesis Representation**.

Input format:
Each line contains an operation represented by characters from 'i', 'and 'e' followed by at most one integer.

- Character 'i' is followed by an integer separated by space. In this operation, a node with this integer as the key is created and inserted into T.
- Character 'p' is to print the **Parenthesis Representation** of the tree T.
- Character 'e' is to 'exit' from the program.

Output Format:
The output is the space-separated Parenthesis Representation of the tree T.

Sample Input :
```
i 4
i 3
i 9
i 5
i 6
i 1
i 8
p
e
```

Output :
```
( 4 ( 3 ( 5 ( ) ( ) ) ( 6 ( ) ( ) ) ) ( 9 ( 1 ( ) ( ) ) ( 8 ( ) ( ) ) ) )
```
(2 marks)

3. A Professor planned to conduct a viva voce as part of the data structures course. So he arranged students of the class based on their marks of the previous exams. His arrangement is as follows: He found the median mark for the previous exam, then he compared each student's mark with median and classified them to below average and above average. Then his selection of calling out the candidates for attending viva was minimum and maximum mark students alternatively like Min, Max, Min, Max (Assume that marks are not duplicated). And he gave his selection procedure the name MinMax Traversal. Choose a suitable data structure (BST) to implement this. The roll number of the median mark will be the root node. Also, design an appropriate node structure for this problem. Your program should contain a min-max traversal function.

Sample Input & Output:

<u>Input format:</u>
The first line denotes the number of students (n) in the class
The next n line contains the roll number and previous exam marks separated by space

Output format:
The order of roll number, a mark called for viva as shown in the sample output

Input

```
9
1 12
2 9
3 8
7 11
5 10
6 14
4 13
8 15
9 16
```

Output
(Roll number,Mark)
(3,8) ,(9,16) ,(2,9), (8,15), (5,10),(6,14), (7,11),(4,13),(1,12) (2 marks)

4. Given a binary tree and a node, write a program to print all cousins of a given node. In a binary tree, two nodes are cousins of each other if they are at the same level and have different parents. Note that siblings should not be printed.

Example:

```
Input:   11
          /  \
       12     13
      /  \     / \
    14    5   16    7
   / \    / \  / \    / \
  8   9 10 4 2 3 15 6
```

Find the cousins of node: 10
Output: 8,9,2,3,15,6

(2 marks)

5. “A hedge fund is an investment vehicle that caters to high-net-worth individuals, institutional investors, and other accredited investors.” The *hedge fund managers* maintain the funds of such wealthy individuals, pool their money together, and try to beat average market

returns. These managers may collude together *(legally/illegally),* often using aggressive strategies in an effort to produce the maximum positive returns for the investors, and they typically get paid based on performance.

We can represent the network of connections, (due to collusion), among *hedge fund managers* using a **Binary Tree**. Each node in the binary tree identifies a manager using a unique **ID** that starts with 'M' followed by three digits from the closed set [0,9]. For example, "M123", "M091", "M000", etc; are valid IDs for the *hedge fund managers*. The second variable of interest is the **Assets** of a *hedge fund manager,* an integer value scaled to Million$, which deals with the total amount of fund he/she manages. For example, the assets of a manager with ID "M123" could be 12 Million$, the assets of a manager with ID "M091" could be 100 Million$, and so on. A given manager X may collude with a maximum of three managers, and in the binary tree representation, they could be represented as the left child or the right child, or as the parent of manager X.

Suppose that set Y contains all managers who are the left-descendants of manager X, and set Z contains all managers who are the right-descendants of manager X, then, the **Equity** handled by manager 'X' is given by the following equation:

$$\textbf{Equity(X)} = (\sum \textbf{Assets(Z)} - \sum \textbf{Assets(Y)})$$

In other words, **Equity(X)** is the sum of the assets of all managers in X's right sub-tree *minus* the sum of the assets of all managers in X's left sub-tree.

The *hedge fund manager* X is said to have colluded **Legally** if the following two conditions are satisfied:

1. Equity handled by manager X becomes equal to his/her Assets handled, *i.e.,*

   $$\textit{Equity(X)} = \textit{Assets(X)}$$

2. The sub-tree S^X rooted at node X (or 'manager X') should be a **Binary Search Tree**, *i.e.,* all managers in the left sub-tree of any node 'N' in S^X should have their assets strictly less than the assets of N and all managers in the right sub-tree of N should have their assets greater than or equal to the assets of N.

NOTE: We can calculate Equity(X) for a manager X, if and only if, manager X has both its left child and right child. Otherwise, if any of the children of X is missing, we cannot calculate Equity(X), and hence we cannot label X to have colluded legally.

Given the unique ID and assets of all the *hedge fund managers* in a network, write a program to find the managers who have colluded legally. Your program should include the functions given below:

- create_BT(): Reads the manager ID and assets from the given parenthesis notation of the binary tree, and creates the corresponding BT, say BT 'T'.
- find_legal(T): Using the BT 'T' created with the help of the create_BT() function**, find and print the manager IDs** of all the *hedge fund managers,* who have colluded legally, **in their post-order traversal form.**

Input Format

The input contains the Parenthesis Representation of the binary tree containing the details of managers, defined recursively, in the following manner.

- The string ( ) represents an empty tree. Note that the open and close parenthesis is separated using a space.
- The string (M_ID *a* LTree RTree) represents a tree whose root node is a manager with Manager ID M_ID and Assets *a*, LTree is the left subtree of the root node in Parenthesis Representation and RTree is the right subtree of the root node in Parenthesis Representation, respectively.

Output Format

The output contains the Manager IDs of all the *hedge fund managers,* who have colluded legally, in the order in which they appear in their post-order traversal form, separated by a space. If no manager is found to have colluded legally, then return -1.

Sample Input0:

```
( M113 2 ( M112 1 ( M229 0 ( ) ( ) ) ( M321 1 ( ) ( ) ) ) ( M029 2 ( ) ( M123 2 ( ) ( ) ) ) )
```

Sample Output0:

```
M112 M113
```

Sample Input1:

```
( M103 8 ( M112 3 ( ) ( ) ) ( M929 11 ( M113 9 ( ) ( ) ) ( M007 20 ( ) ( ) ) ) )
```

Sample Output1:

```
M929
```

Sample Input2:

```
( M103 8 ( M112 3 ( ) ( ) ) ( M929 9 ( M113 11 ( ) ( ) ) ( M007 20 ( ) ( ) ) ) )
```

<u>Sample Output2</u>: (2 marks)

```
-1
```