National Institute of Technology Calicut
Department of Computer Science and Engineering
Fourth Semester B. Tech.(CSE)-Winter 2021
CS2094D Data Structures Laboratory
Assignment #1

Submission deadline (on or before): 08.02.2021, 09:00 AM

Policies for Submission and Evaluation:

- Programs should be written in C language and compiled using C compiler in Linux platform.
- 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.
- 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 related to the lab course can lead to awarding an F grade in the course.

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: $ASSG1\_BxxyyyyCS\_LAXMAN.zip$). 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: $ASSG1\_BxxyyyyCS\_LAXMAN\_1.c$). 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 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: http://cse.nitc.ac.in/sites/default/files/Academic-Integrity_new.pdf.

QUESTIONS

1. A Binary Tree is a rooted 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 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 a binary tree T, do a level order traversal of the given tree T. If we find a node whose left child is empty, make new key k as left child of the node. Else if we find a node whose right child is empty, make the new key k as right child. i.e., keep traversing the tree until we find a node whose either left or right child is empty.
   - 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 a character from 'i', 'p'and 'e' followed by at most one integer. The integers, if given, are in the range $[-10^6, 10^6]$.
   - Character 'i' 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 '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

   Sample Output :

   (4 (3 (5 ( ) ( )) (6 ( ) ( ))) (9 (1 ( ) ( )) (8 ( ) ( ))))

2. In a binary tree, two nodes are *cousins* of each other if they are at the same level and have different parents. Given a binary tree T and a node x, write a program to print all *cousins* of the given node x. If x has no *cousins*, print -1. Assume that T contains no duplicate nodes. Note that siblings of x should not be printed.

Input format:

- The integers, if given, are in the range $[-10^6, 10^6]$.
- First line of the input contains space separated Parenthesis Representation of the tree T.
- Second line contains the integer k.

Output format:

- The output is the cousins of the node x with input integer k as key separated by a space. If x has no cousins, print -1.

Sample Input 1 :
(4 (3 (5 ( ) ( )) (6 ( ) ( ))) (9 (1 ( ) ( )) (8 ( ) ( ))))
6
Sample Output 1 :
1 8

Sample Input 2 :
(4 (3 (5 ( ) ( )) (6 ( ) ( ))) (9 ( ) ( )))
6
Sample Output 2 :
-1

3. Given a binary tree T, write a program to count the total number of sub-trees of T which is also a Binary Search Tree (BST) and the sum of all the nodes of that BST is equal to the given sum k. If no such sub-trees are present, print -1.

Input Format:

- The integers, if given, are in the range $[-10^6, 10^6]$.
- First line of the input contains space separated Parenthesis Representation of the tree T.
- Second line of the input contains the integer value k.

Output Format:

- Print the total number of BST whose sum is equal to given k. If no such sub-trees are present, print -1.

Sample Input 1:
(5 (4 (3 ( ) ( )) (6 ( ) ( ))) (3 (1 ( ) ( )) (9 ( ) ( ))))
13
Sample Output 1:
2

Sample Input 2:
(5 (4 (3 ( ) ( )) (3 ( ) ( ))) (6 (1 ( ) ( )) (9 ( ) ( ))))
2
Sample Output 2:
-1

4. 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 $y.key \leq x.key$. 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 level*, *find minimum*, *find maximum*, *predecessor*, *successor* and *traversals*(inorder, preorder and postorder) on T. Input should be read from 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', 'l', 'm', 'x', 'r', 'u', 'i', 'p', 't' 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.
   - LEVEL(T, k) - searches for a node with key k in T, and returns the level of the 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.
   - PREDECESSOR(T, y) - searches for a node with key y in T, and returns a pointer to a node which is predecessor of the node with key y if one exists; otherwise, it returns NIL.
   - SUCCESSOR(T, y) - searches for a node with key y in T, and returns a pointer to a node which is successor of the node with key y if one exists; otherwise, it returns NIL.
   - INORDER(T) - performs recursive inorder traversal of the BST T and prints the data in the nodes of T in inorder.
   - PREORDER(T) performs recursive preorder traversal of the BST T and prints the data in the nodes of T in preorder.
   - POSTORDER(T) performs recursive postorder traversal of the BST T and prints the data in the nodes of T in postorder.

   Input format:

   - Each line contains a character from 'a', 'd', 's', 'l', 'm', 'x', 'r', 'u', 'i', 'p', 't' or 'e' followed by at most one integer. The integers, if given, are in the range $[-10^6, 10^6]$.
   - 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 *'l'* is followed by an integer separated by space. This operation is to find the level of 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 *'r'* is followed by an integer separated by space. This operation is to find the predecessor of the node with this integer as key in T.
- Character *'u'* is followed by an integer separated by space. This operation is to find the successor of the node with this integer as key in T.
- Character *'i'* is to perform inorder traversal of T.
- Character *'p'* is to perform preorder traversal of T.
- Character *'t'* is to perform postorder traversal 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 key is not present in T, then print -1.
- For option '*l*', if the key is present in T, then print its level. If 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.
- For option '*r*', if the key is present in T, then print its predecessor. If key is not present in T, then print -1.
- For option '*u*', if the key is present in T, then print its successor. If key is not present in T, then print -1.
- For option *'i'*, print the data in the nodes of T obtained from inorder traversal.
- For option *'p'*, print the data in the nodes of T obtained from preorder traversal.
- For option *'t'*, print the data in the nodes of T obtained from postorder traversal.

Sample Input :

```
a 25
a 13
a 50
a 45
a 55
a 18
l 50
l 19
m
x
r 25
u 30
u 25
i
p
t
s 10
```

s 25
d 55
d 13
d 10
d 25
i
s 25
e

Sample Output :

2
-1
13
55
18
-1
45
13 18 25 45 50 55
25 13 18 50 45 55
18 13 45 55 50 25
-1
1
55
13
-1
25
18 45 50
-1

5. Given a Binary Search Tree(BST) T and a key value k. Write a program that implements the following function:

   KSMALLEST(T, k) : Takes as input the root of a BST T and an integer k, such that $k \leq n$, where n is the number of nodes in T and returns the k-th smallest value in the BST T. Note that n is not part of the input. **Input Format:**

   - The integers, if given, are in the range $[-10^6, 10^6]$.
   - First line of the input contains space separated PARENTHESIS REPRESENTATION of the tree T.
   - Second line contains the integer k.

   Output Format:

   - Print the kth smallest element in the BST.

   Sample Input 1:

   (8 (5 (3 ( ) ( )) (6 ( ) ( ))) (13 (10 ( ) ( )) (19 ( ) ( ))))
   5

   Sample Output 1:

   10

6. Given a list of n numbers sorted in ascending order, develop a program to determine the order in which these elements should be inserted into a BST such that the tree construction should be completed in $O(n \log n)$ time.(ie. the tree should be height balanced). Print the BST created by maintaining this property. Also find the level-wise sum of the elements in the tree, starting from root to leaf.

Note: To insert n nodes into a BST you need to make at least $O(\log n)$ comparisons. Hence the time complexity for the construction of BST with n elements will be minimum as $O(n \log n)$.

Input Format:

- The integers, if given, are in the range $[-10^6, 10^6]$.
- Read the number of elements n in the tree.
- Second line of the input should contain space separated integers(n in number) in ascending order.

Output Format:

- First line of the output should contain space separated Parenthesis Representation of the tree T.
- Next line of the output should contain level sums starting from root ($\lfloor \log n \rfloor + 1$ entries) separated by single space.

Sample Input 1:

```
7

18 20 24 30 36 50 51
```

Sample Output 1:

```
( 30 ( 20 ( 18 ( ) ( ) ) ( 24 ( ) ( ) ) ) ( 50 ( 36 ( ) ( ) ) ( 51 ( ) ( ) ) ) )

30 70 129
```