National Institute of Technology Calicut
Department of Computer Science and Engineering
Fourth Semester B. Tech.(CSE)-Winter 2022-23
CS2094D Data Structures Laboratory
Assignment #1

Submission deadline (on or before): 25.01.2023, 5:00 PM

Policies for Submission and Evaluation:

- You must submit your assignment in the Eduserver course page, on or before the submission deadline.
- Ensure that your programs will compile and execute without errors using gcc compiler.
- 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. Given the PARENTHESIS REPRESENTATION of a binary tree T, a node with key value x and a positive integer $k<n$(where n is the number of nodes in the binary tree), print all descendant of x at a distance of k from x in T. Assume unique key values.

   Input format:

   - First line of the input contains space separated PARENTHESIS REPRESENTATION of the tree T with key values $\in [1, 10^6]$.
   - Second line contains an integer x.
   - Third line contains an integer $k \in [1, 10^3]$.

   Output format:

   - The output consists of a single line with space-separated key values.

   Sample Input 1:
   ```
   ( 2 ( 7 ( 10 ( ) ( ) ) ( 6 ( 5 ( ) ( ) ) ( 11 ( ) ( ) ) ) ) ( 9 ( ) ( 3 ( ) ( ) ) ) )
   7
   2
   ```

   Sample Output 1:
   ```
   5 11
   ```

2. Given the PARENTHESIS REPRESENTATION of a binary tree T, write a program to count the number of nodes whose grandparent has a child as a leaf node in a given binary tree.

   Input format:

   - First line of the input contains space separated PARENTHESIS REPRESENTATION of the tree T with key values $\in [1, 10^6]$.

   Output format:

   - The output consists of a single integer value.

   Sample Input 1:
   ```
   ( 10 ( 5 ( 8 ( 3 ( ) ( ) ) ( ) ) ( 7 ( ) ( ) ) ) ( 9 ( ) ( ) ) )
   ```

   Sample Output 1:
   ```
   3
   ```

   Sample Input 2:
   ```
   ( 50 ( 90 ( 30 ( ) ( ) ) ( ) ) ( 40 ( 80 ( ) ( ) ) ( 10 ) ( ) ( ) ) )
   ```

   Sample Output 2:
   ```
   0
   ```

3. Given the PARENTHESIS REPRESENTATION of a binary tree T and a sum, write a program to print the number of subtrees whose sum of the key values is equal to the given sum.

   Input format:

- First line of the input contains space separated PARENTHESIS REPRESENTATION of the tree T with key values $\in [1, 10^6]$.
- Second line contains an integer representing sum.

Output format:

- The output consists of a single integer value.

Sample Input 1:

```
( 1 (3 ( 5 ( ) ( ) ) ( 9 ( ) ( ) ) ) ( 6 ( 11 ( ) ( ) ) ( ) ) )

17
```

Sample Output 1:

```
2
```

Sample Input 2:

```
( 5 ( 2 ( 1 ( ) ( ) ) ( 3 ( ) ( ) ) ) ( 12 (9 ( ) ( ) ) (21 ( ) ( ) ) ) )

7
```

Sample Output 2:

```
0
```

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* and *right* that point to the nodes corresponding to its left child and its right child, respectively. If a child is missing, the appropriate attribute contains the value 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*' or '*t*' from the console and calls the sub-functions appropriately until character '*e*' is entered for exiting the program.
   - 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 key in the nodes of T in inorder.
- Preorder(T) performs recursive preorder traversal of the BST T and prints the key in the nodes of T in preorder.
- Postorder(T) performs recursive postorder traversal of the BST T and prints the key 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 single 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 single 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 single space. This operation is to find the node with this integer as key in T.
- Character '*l*' is followed by an integer separated by single 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 single 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 single 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 key in the nodes of T obtained from inorder traversal.
- For option *'p'*, print the key in the nodes of T obtained from preorder traversal.
- For option *'t'*, print the key 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. The highest path in a binary search tree is the path from the root to a leaf node whose sum is greater than that of all other paths. Given the PARENTHESIS REPRESENTATION of a binary search tree(BST) T, write a program that prints the highest path and its corresponding sum. If there are multiple paths with the same sum, the program should print the shortest path among them, where

the shortest path is defined as the one with the least number of nodes.

Input format:

- First line of the input contains space-separated PARENTHESIS REPRESENTATION of the tree T with key values $\in [1, 10^6]$.

Output format:

- The output consists of a single line: $\{L, R\}^*$ sum, where $\{L, R\}^*$ is a string representing the path and sum is an integer.

  Note: If the BST contains only one node(root), there is no need to print the path.

Sample Input 1:
```
( 100 ( 80 ( 70 ( 50 ( 40 ( ) ( ) ) ( 60 ( ) ( ) ) ) ( 75 ( ) ( ) ) ) ( 95 ( ) ( ) ) ) ( 102 ( ) ( ) ) )
```

Sample Output 1:
```
LLLR 360
```

Sample Input 2:
```
( 50 ( ) ( ) )
```

Sample Output 2:
```
50
```

Sample Input 3:
```
( 50 ( 20 ( 10 ( ) ( ) ) ( 40 ( ) ( ) ) ) ( 60 ( ) ( ) ) )
```

Sample Output 3:
```
R 110
```

6. Given the PARENTHESIS REPRESENTATION of a binary search tree(BST) T and two key values x and y ($x \neq y$) that specify the range of key values, print the key values of the nodes that are in the given range(inclusive of x and y).

Input format:

- First line of the input contains space separated PARENTHESIS REPRESENTATION of the tree T with key values $\in [1, 10^6]$.
- Second line of the input contains two key values, x and y, separated by space where $x, y \in [1, 10^6]$.

Output format:

- The output consists of a single line with space-separated key values.

Sample Input 1:
```
( 8 ( 5 ( ) ( 6 ( ) ( ) ) ) ( 10 ( ) ( 11 ( ) ( ) ) ) )
6 10
```

Sample Output 1:
```
6 8 10
```

Sample Input 2:
(50 (40 (30 ( ) ( )) (45 ( ) ( ))) (60 ( ) ( )))
45 50

Sample Output 2:
45 50

7. Given the Parenthesis Representation of a binary search tree(BST) T and a node with key value k, write a program to print all the ancestors of k in T. Assume unique key values.

   Input format:

   - First line of the input contains space separated Parenthesis Representation of the tree T with key values $\in [1, 10^6]$.
   - Second line of the input contains a node with the key value k in T where $k \in [1, 10^6]$.

   Output format:

   - The output consists of space-separated key values of order starting from parent.

   Sample Input 1:
   (16 (13 (12 ( ) ( )) (14 ( ) ( ))) (23 (20 ( ) ( )) (32 ( ) ( )))))
   14

   Sample Output 1:
   13 16

   Sample Input 2:
   (10 (5 (4 (2 ( ) ( )) ( )) (7 ( ) ( ))) (12 ( ) ( )))
   2

   Sample Output 2:
   4 5 10