National Institute of Technology Calicut
Department of Computer Science and Engineering
Third Semester B. Tech.(CSE)
CS2092D Programming Laboratory
Assignment #8
Submission deadline (on or before): 02/11/2023, 11:55 PM

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: *ASSG8_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: *ASSG8_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.

General Instructions

- Programs should be written in C language.
- Check your programs with sufficiently large values of inputs with in the range as specified in the question.
- Global and/or static variables should not be used in your program.

QUESTIONS

1. Write a program to create a **Special Binary Search Tree** T with distinct keys (with values in the range $[1, 10^6]$) and perform related operations. A Special BST is organized as a binary tree in which each node is an object that contains a *key* value. In addition to a *key*, each node contains three attributes *left*, *right*, and *parent* that point to the nodes corresponding to its left child, right child, and 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 special binary search tree are always stored in such a way as to satisfy the following property:

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

   Your program should include the following functions.

- MAIN() - repeatedly reads a character 'a', 's', 'x', 'n', 'i', 'p', 't' or 'e' from the console and calls the sub-functions appropriately until character 'e' is entered.
- CREATE-NODE(K) - creates a new node of the tree with the 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, K) - inserts the node with key value 'k' into the BST T.
  Note: The caller of this function is assumed to create the node using the Create-Node() function with key value 'k'.
- 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.
- FIND-MAX(T) - return the maximum among all the key values in T.
- FIND-MIN(T) - return the minimum among all the key values in T.
- INORDER(T) - performs recursive inorder traversal of the Special BST T and prints the key value in the nodes of T in inorder.
- PREORDER(T) - performs recursive preorder traversal of the Special BST T and prints the key value in the nodes of T in preorder.
- POSTORDER(T) - performs recursive postorder traversal of the Special BST T and prints the key value in the nodes of T in postorder.

Input format:

- Each line contains a character from 'a', 's', 'x', 'n', 'i', 'p', 't' or 'e' followed by at most one integer. The integers, if given, are in the range $[1, 10^6]$.
- 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 by calling the function INSERT().
- Character 's' is followed by an integer separated by space. This operation is to find the node with this integer as key in T by calling the function SEARCH().
- Character 'x' is to call the function FIND-MAX().
- Character 'n' is to call the function FIND-MIN().
- Character 'i' is to perform inorder traversal of T by calling the function INORDER().
- Character 'p' is to perform preorder traversal of T by calling the function PREORDER().
- Character 't' is to perform postorder traversal of T by calling the function POSTORDER().
- Character 'e' is to exit the program.

Output Format:

- The output (if any) of each command should be printed on a separate line.
- For option 's', if the key is present in T, then print 'F'. If the key is not present in T, then print 'N'. Print NIL if T is empty.
- For option 'x', print the maximum among all the key values in T.
- For option 'n', print the minimum among all the key values in T.
- For option 'i', print the key value in the nodes of T obtained from inorder traversal.
- For option 'p', print the key value in the nodes of T obtained from preorder traversal.
- For option 't', print the key value in the nodes of T obtained from postorder traversal.

Note: Print NIL if T is empty. (**Note:** NIL means nothing is to be printed.)

Sample Input:

s 50
x
n
i

p
t
a 25
a 13
a 50
a 45
a 55
a 18
x
n
i
p
t
s 10
s 25
e

Sample Output:
55
13
55 50 45 25 18 13
25 50 55 45 13 18
55 45 50 18 13 25
N
F

2. Write a program to create a **Special Binary Search Tree T** (as specified in question1) with distinct keys (with values in the range $[1, 10^6]$) and perform the below specified operations. Reuse the definitions and required functions from Question1 to complete this question. Input should be read from console and output should be shown in console.

   Your program should include the following functions.

   - MAIN() - repeatedly reads a character 'a', 'd', 'c', 'r', 'p' or 'e' from the console and calls the sub-functions appropriately until character 'e' is entered.
   - CREATE-NODE(K) - creates a new node of the tree with the 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, K) - inserts the node with key value 'k' into the BST T.
     Note: The caller of this function is assumed to create the node using the Create-Node() function with key value 'k'.
   - DELETE(T, K) - deletes the node with key value k from the BST T . If a node with the input key is not present in T , then print -1. Note: The caller of this function is assumed to invoke Search() function to locate the node with key value k.
   - SUCCESSOR(T, K) - finds the successor of the node with key k in the BST T and prints the key value of the successor node. Successor of the node I is the node that comes after node I in the inorder traversal of the tree T.
   - PREDECESSOR(T, K) - finds the predecessor of the node with key k in the BST T and prints the key value of the predecessor node. Predecessor of the node I is the node that comes before node I in the inorder traversal of the tree T .
   - PREORDER(T) - performs recursive preorder traversal of the BST T and prints the key value in the nodes of T in preorder.

   Note1 : If a node with the input key is not present in T , then print -1. **Note 2** : If the T is empty, then print -1.

Note 1 : Successor of the node I is the node that comes after node I in the inorder traversal of the tree T
Note 2 : Predecessor of the node I is the node that comes before node I in the inorder traversal of the tree T

Input format:

- Each line contains a character from 'a', 'd', 'c', 'r', 'p', or 'e' followed by at most one integer. The integers, if given, are in the range $[1, 10^6]$.
- 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 by calling the function INSERT().
- 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 calling the function DELETE().
- Character 'c' is followed by an integer separated by space. This operation is to find successor of the node with this integer as key in T by calling the function SUCCESSOR().
- Character 'r' is followed by an integer separated by space. This operation is to find predecessor of the node with this integer as key in T by calling the function PREDECESSOR().
- Character 'p' is to perform preorder traversal of T by calling the function PREORDER().
- Character 'e' is to exit 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 'c', if the successor is present in T , then print the data in the successor node. If successor is not present in T, then print -1.
- For option 'r', if the predecessor is present in T , then print the data in the predecessor node. If predecessor is not present in T, then print -1.
- For option 'p', print the key value in the nodes of T obtained from preorder traversal. Print -1 if T is empty.

Sample Input:
a 50
d 20
a 30
a 60
a 20
a 40
c 30
r 40
p
e

Sample Output:
-1
20
50
50 60 30 40 20

3. The PARENTHESIS REPRESENTATION of a binary search 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 and right-subtree is the right subtree of the root node following the **binary-search-tree** property in PARENTHESIS REPRESENTATION.

Write a program to create a binary search tree T, delete an **existing** node from the tree, and print T in Parenthesis Representation. Your program should contain the following functions:

- INSERT(T, k) - inserts the element 'k' to the BST T.
- DELETE(T, k) - deletes an existing element 'k' from the BST T and place inorder successor.
- PRINT(T) - that should take as input a pointer to the root node of the BST and print the tree in its PARENTHESIS REPRESENTATION.

Input format:

- Each line contains a character from 'i', 'd', 'p' and 'e' followed by at most one integer. The integers, if given, are in the range $[1, 10^6]$.
- Character 'i' is followed by an integer separated by a single space. A node with this integer as a key is created and inserted into T in this operation.
- Character 'd' is followed by an integer separated by a single space. A node with this integer as a key is deleted from the T in this operation.
- Character 'p' is to print the Parenthesis Representation of the BST T.
- Character 'e' is to exit the program.

Output format:

- The output is the space-separated Parenthesis Representation of the BST T. **Note**: The integer values, '(' and ')', are separated by exactly one space.

Sample Input:

```
p
i 10
p
i 20
p
i 5
d 10
p
i 40
i 15
i 25
p
```

Sample Output:

```
( )
( 10 ( ) ( ) )
( 10 ( ) ( 20 ( ) ( ) ) )
( 10 ( 5 ( ) ( ) ) ( 20 ( ) ( ) ) )
( 20 ( 5 ( ) ( ) ) ( ) )
( 20 ( 5 ( ) ( 15 ( ) ( ) ) ) ( 40 ( 25 ( ) ( ) ) ( ) ) )
```

4. Write a program to construct a binary tree T (Not necessarily a Binary Search Tree) from the given Parenthesis Representation (as described in Question 3). Your program should contain the following functions:

- BUILD-TREE(S) - A recursive function that builds a binary tree T from the given Parenthesis Representation string S.
- PRINT(T) - A function that take as input a pointer to the root node of the tree T and print the inorder traversal of T .

Input format:

- The first and only line of input contains a valid Parenthesis Representation of a binary tree, represented as a string S, length(S)$\leq$ 103 . The keys which are integers are in the range $[1, 10^6]$.

Output format:

- Prints the inorder traversal of the given binary tree where the integer keys are printed in one line separated by single space.

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