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):

- 07.11.2022, 11:55 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.
- Detection of ANY malpractice related to the lab course can lead to awarding an F grade in the course.

Naming Conventions for Submission

- The source codes must be named as

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>_<PROGRAM-NUMBER>.c`

  (For example: *ASSG1_BxxyyyyCS_LAXMAN_1.c*).

  Create separate files for each question and place all of them in a single folder named

  `ASSG<NUMBER>_<ROLLNO>_<FIRST-NAME>`

  (Example: *ASSG1_BxxyyyyCS_LAXMAN*). Then Zip the **folder**.

- Submit the single ZIP (.zip) file you have created(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.

  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: https://minerva.nitc.ac.in/?q=node/650.

General Instructions

- Programs should be written in C language and compiled using gcc compiler. **Submit the solutions to the questions through the submission link in Eduserver**.
- 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.
- You shouldn't change the function prototype specified in the question.

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*, each node contains three attributes `left`, `right`, and `p` 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 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 with distinct keys (with values in the range $[1, 10^6]$) and perform related operations. 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', '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 *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 BST T and prints the key value in the nodes of T in inorder.
    - PREORDER(T) performs recursive preorder traversal of the BST T and prints the key value in the nodes of T in preorder.
    - POSTORDER(T) performs recursive postorder traversal of the 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' from 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 `Found`. If key is not present in T, then print `NotFound`.
    - For option 'x', print the maximum among all the *key* values in T. Print `NIL` if T is empty.
    - For option 'n', print the minimum among all the *key* values in T. Print `NIL` if T is empty.

- For option *'i'*, print the key value in the nodes of T obtained from inorder traversal. Print `NIL` if T is empty.
- For option *'p'*, print the key value in the nodes of T obtained from preorder traversal. Print `NIL` if T is empty.
- For option *'t'*, print the key value in the nodes of T obtained from postorder traversal. Print `NIL` if T is empty.

Sample Input:

```
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
13 18 25 45 50 55
25 13 18 50 45 55
18 13 45 55 50 25
NotFound
Found
```

2. Write a program to create a BINARY SEARCH TREE T with distinct keys (with values in the range $[1, 10^6]$) and perform the below specified operations. Reuse the definitions and required functions from Question 1 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 *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 CREATENODE() function with *key* value k. When deleting a node with two children, choose inorder successor.
   - DELETE(T, k) - deletes the node with *key* value k from the BST T.
     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.
   - 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.
   - PREORDER(T) performs recursive preorder traversal of the BST T and prints the key value in the nodes of T in preorder.

   Input format:

   - Each line contains a character from '*a*', '*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 '*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’ 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 `NotFound`.
- 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 `NotFound`.
- 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 `NotFound`.
- For option *‘p’*, print the key value in the nodes of T obtained from preorder traversal. Print `NIL` if T is empty.

Sample Input:

```
a 25
a 13
a 50
a 45
a 55
a 18
c 25
c 55
r 45
r 13
d 55
d 13
d 10
d 25
p
e
```

Sample Output:

```
45
NotFound
25
NotFound
55
13
NotFound
25
45 18 50
```

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 and print T in PARENTHESIS REPRESENTATION. Your program should contain the following functions:

   - INSERT(T, k) - inserts the element k to the BST T.
   - 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’, ‘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 single 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 BST T.
- Character ‘e’ is to ‘exit’ from 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 4
p
i 3
p
i 9
i 5
i 6
i 1
i 8
p
e
```

Sample Output:

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

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 postorder 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) $\leq 10^3$). The keys which are integers are in the range $[-10^6, 10^6]$.

Output format:

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

Sample Input:

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

Sample Output:

```
5 6 3 1 8 9 4
```