G. H. Raisoni College Of Engineering And Management, Wagholi Pune

2021- 2022

Assignment no :- 8

Department	CE [SUMMER 2022 (Online)]		
Term / Section	III/B	Date Of **submission**	13-12-2021
Subject Name /Code	Data Structures and Algorithms/ UCSL201/UCSP201		
Roll No.	SCOB77	Name	Pratham Rajkumar pitty
Registration Number	2020AC0E1100107		

Experiment No 8

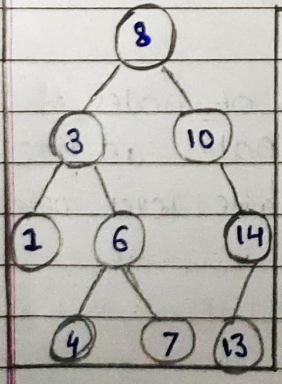
Aim: Beginning with an empty binary search tree, construct binary search tree by inserting the values in the order given. After constructing a binary tree -

(i) Inserting new node
(ii) Find number of nodes in longest path
(iii) Minimum data values found in the tree
(iv) Change a tree so that the roles of the left and right pointers are swapped at every node
(v) Search a value

Theory :→

Binary search tree is a node-based binary tree data structure which has the following properties:

- The left subtree of a node contains only nodes with keys lesser than the node's key.
- The right subtree of a node contains only nodes with keys ~~lesser~~ greater than the node's key.
- The left & right subtree of a node must also be a binary search tree.



ALGORITHMS:→

- Create and Insert:

1. Accept a random order of numbers from the user. The first number would be inserted at the root node.
2. There after, every number is compared with the root node. If less than or equal to the data in the root node, proceed left of the BST. else proceed right.

3. Perform this till you reach a null pointer, the place where the present data is to be inserted
4. Allocate a new node and assign the data in this node allocate pointers appropriately.

Height of BST:

1. This algorithm is based on the idea of storing the nodes of every level of the BST in a dynamic queue (link list). It is also simultaneously useful to print the tree level wise.

2. Initialize the contents of the with the root of the BST. The counter no_of_nodes_in_current_level=1 & the level_accessed=1.

3. Access no_of_nodes_in_current_level from the link list and add all their children to the list at the end and simultaneously keep track of the number of nodes accessed in the next level in a variable which at the end is assigned back to no_of_nodes_in_current_level. Also increment level_accessed, indicating one more level accessed in the BST.

4. Continue Step3 repeatedly till no_of_nodes_in_current level is 0, which means no more nodes in the next level. The value stored in the variable level_accessed is the height of the BST.

Leaf Nodes of BST:

1. There are many algorithms to find the leaf nodes of a

BST. The one considered here is based on the idea that one could do a simple inorder traversal of the BST and just before printing the data as one normally does in an inorder traversal, check if both the left and right nodes are NULL. If so, it means the node under consideration is a leaf node and must be printed.

2. Inorder: The recursive function will receive the root of the tree (or subtree) from where inorder traversal is to be initiated. Algorithm is to proceed left, which is this case is to call the same function with the left child of the node, print the data if both left and right pointer are NULL and then proceed right, which in this case is to call the same function with the right child of the node.

3. Thus all the leaf nodes of the BST are printed.

Mirror of Tree:

1. Following is a algorithm of a recursive function to find mirror of the tree. The function Mirror_Tree accepts a pointer to a tree as the parameter. Initially the root node is passed as the parameter.

2. The function begins by checking if the pointer passed is not NULL. If not, allocates a new node. Assigns the data of the ~~original~~ original node to the copied node. Assigns the left child of the new node by calling the function Mirror_Tree with the right child of the original node and assigns the right child of the new node by calling the function mirror_Tree with the left child of the ~~original~~ original node. If the pointer passed is NULL, NULL

is returned by the function else the new node created is returned.

Level wise printing:

1. This algorithm is based on the idea of storing the nodes of every level of the BST in a dynamic queue (link list).
2. Initialize the contents of the list with the root of the BST. The counter no_of_nodes_in_current_level ~~from the level link list~~ _level = 1 and the level_accessed = 1
3. Access no of nodes in current level from the link list. Print the Level Number and all the data of all the nodes of the current level. Simultaneously add all their Children to the list at the end and keep track of the number of nodes accessed in the next level in a variable which at the end is assigned back to no_of_nodes_in_current_level. Also increment level_accessed, indicating one more level accessed in the BST.
4. Continue Step 3 repeatedly till no_of_nodes_in_current_level is 0, which means no more nodes in the next level.

Test conditions

Simple input of random numbers. Display the height of the tree and the leaf nodes. The BST entered could be drawn on a rough page and could check if the height calculated and the leaf nodes printed are correct.

For eg, Enter : 34, 12, 56, 6, 14, 40, 70.

The height of the BST is 3
The leaf nodes are 6, 14, 40 and 70

For mirror Image

Enter : 34, 12, 56, 6, 14, 40, 70.

Level wise printing of original tree

Level 1 : 34
Level 2 : 12, 56.
Level 3 : 6, 14, 40, 70

Level wise printing of the mirror tree

Level 1 : 34
Level 2 : 56, 12
Level 3 : 70, 40, 14, 6

INPUT :→

Enter data (numbers) to be stored in the BST would contain 3 fields : data, left child pointer and right child pointer

OUTPUT :→

The height of the tree and the list of its leaf nodes.

The original and mirror image pointed levelwise.

Program code :-

```
#include <iostream>
using namespace std;
struct node
{
   int data;
   node *L;
   node *R;
};
node *root, *temp;
int count, key;

class bst
{
public:
   void create();
   void insert(node *, node *);
   void disin(node *);
   void dispre(node *);
   void dispost(node *);
   void search(node *, int);
   int height(node *);
   void mirror(node *);
   void min(node *);
   bst()
   {
      root = NULL;
      count = 0;
   }
```

```
};

void bst::create()
{
   char ans;
   do
   {
      temp = new node;
      cout << "Enter the data : ";
      cin >> temp->data;
      temp->L = NULL;
      temp->R = NULL;
      if (root == NULL)
      {
         root = temp;
      }
      else
         insert(root, temp);
      cout << "Do you want to insert more value :(y/n) : " << endl;
      cin >> ans;
      count++;
      cout << endl;
   } while (ans == 'y');
   cout << "The Total no.of nodes are : " << count;
}
void bst::insert(node *root, node *temp)
{
   if (temp->data > root->data)
    {
```

```
        if (root->R == NULL)
        {
            root->R = temp;
        }
        else
            insert(root->R, temp);
    }
    else
    {
        if (root->L == NULL)
        {
            root->L = temp;
        }
        else
            insert(root->L, temp);
    }
}

void bst::disin(node *root)
{
    if (root != NULL)
    {
        disin(root->L);
        cout << root->data << "\t";
        disin(root->R);
        count++;
    }
}
void bst::dispre(node *root)
```

```
{
   if (root != NULL)
   {
      cout << root->data << "\t";
      dispre(root->L);
      dispre(root->R);
   }
}
void bst::dispost(node *root)
{
   if (root != NULL)
   {
      dispost(root->L);
      dispost(root->R);
      cout << root->data << "\t";
   }
}
void bst::search(node *root, int key)
{
   int flag = 0;
   cout << "\nEnter your key : " << endl;
   cin >> key;
   temp = root;
   while (temp != NULL)
   {
      if (key == temp->data)
      {
         cout << "KEY FOUND\n";
         flag = 1;
```

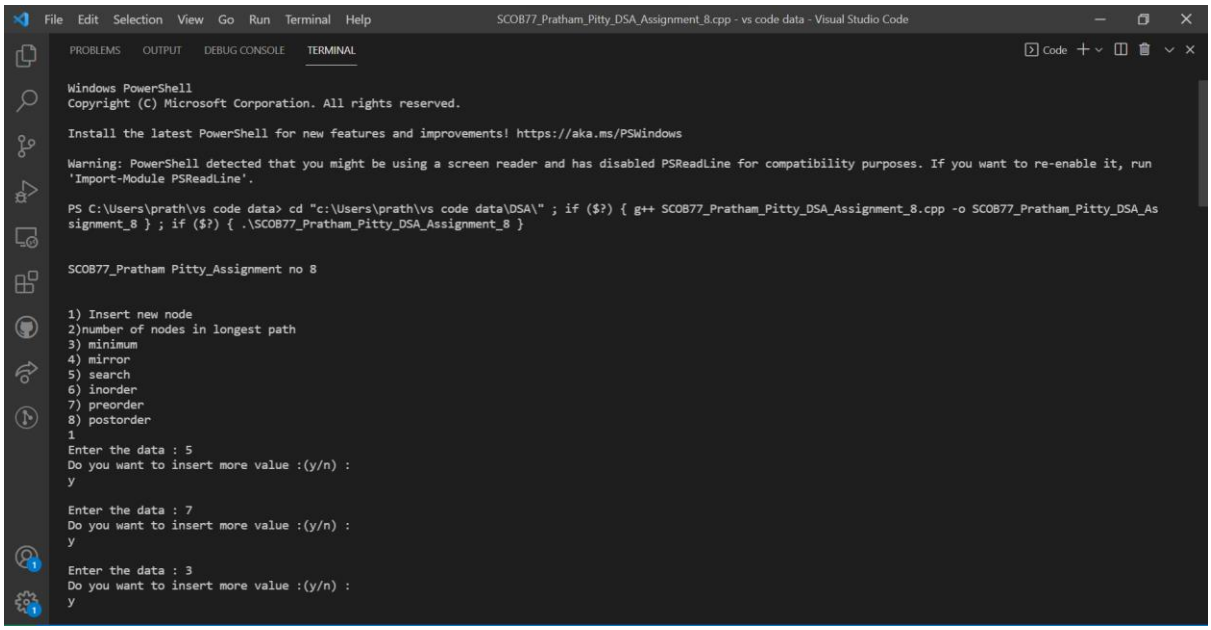
```
            break;
        }
        node *parent = temp;
        if (key > parent->data)
        {
            temp = temp->R;
        }
        else
        {
            temp = temp->L;
        }
    }
    if (flag == 0)
    {
        cout << "KEY NOT FOUND " << endl;
    }
}
int bst::height(node *root)
{
    int hl, hr;
    if (root == NULL)
    {
        return 0;
    }
    else if (root->L == NULL && root->R == NULL)
    {
        return 0;
    }
    cout << endl;
```

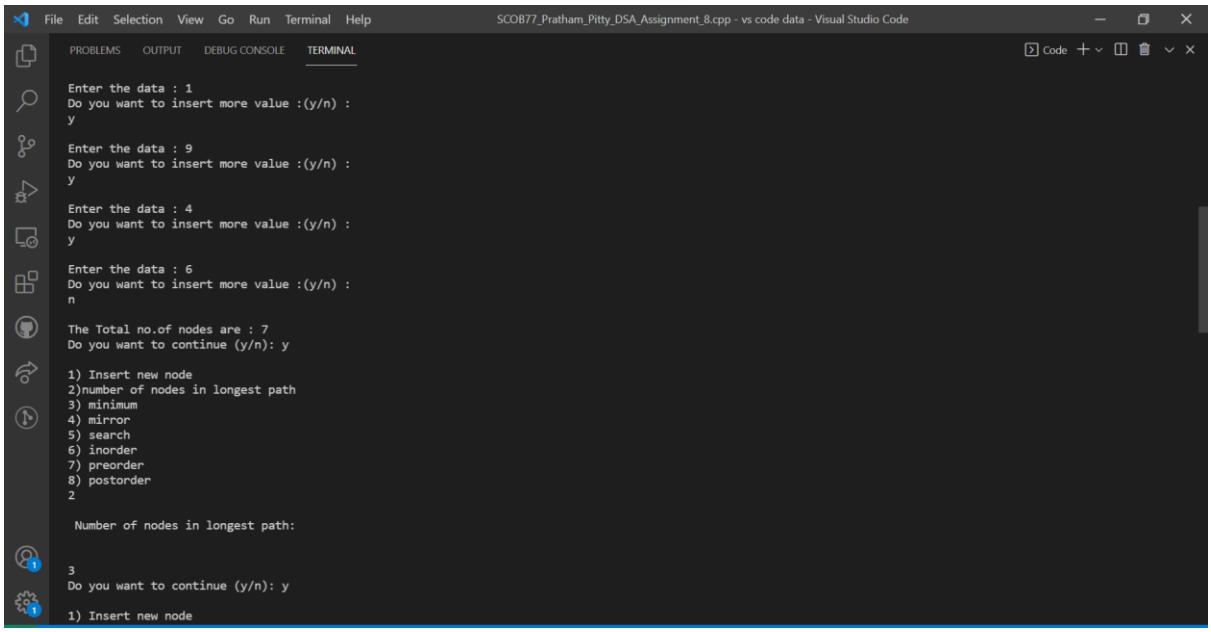
```
    hr = height(root->R);
    hl = height(root->L);
    if (hr > hl)
    {
        return (1 + hr);
    }
    else
    {
        return (1 + hl);
    }
}

void bst::min(node *root)
{
    temp = root;
    cout << endl;
    while (temp->L != NULL)
    {
        temp = temp->L;
    }
    cout << root->data;
}

void bst::mirror(node *root)
{
    temp = root;
    if (root != NULL)
    {
        mirror(root->L);
```

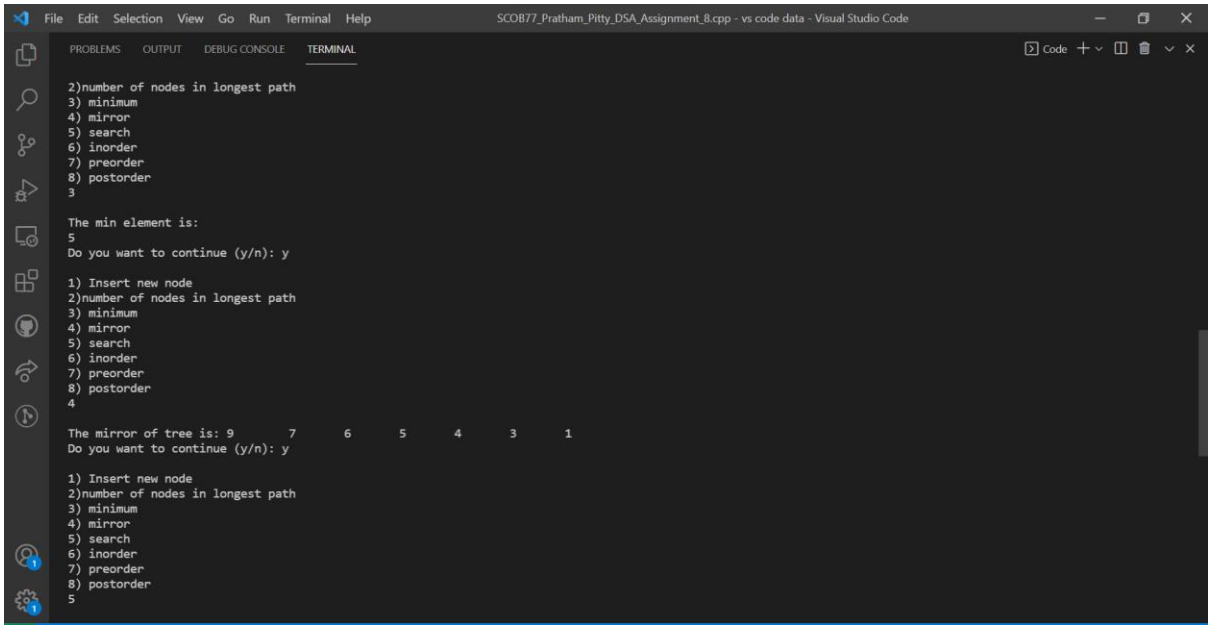
```
        mirror(root->R);
        temp = root->L;
        root->L = root->R;
        root->R = temp;
    }
}
int main()
{
    cout << "\n\nSCOB77_Pratham Pitty_Assignment no 8 \n\n";

    bst t;
    int ch;
    char ans;
    do
    {
        cout << "\n1) Insert new node\n2)number of nodes in longest path\n3) minimum\n4) mirror\n5) search\n6) inorder\n7) preorder\n8) postorder" << endl;
        cin >> ch;
        switch (ch)
        {
        case 1:
            t.create();
            break;
        case 2:
            cout << "\n Number of nodes in longest path: " << (1 + (t.height(root)));
            break;
        case 3:
            cout << "\nThe min element is: ";
            t.min(root);
            break;
```

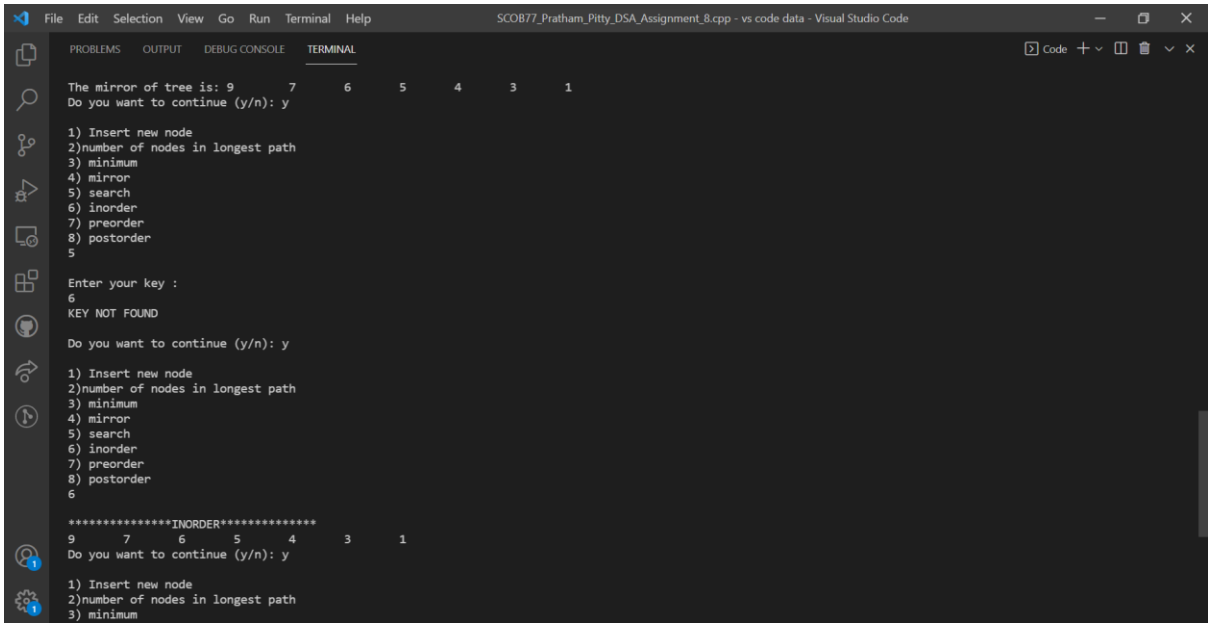
```
        case 4:
            t.mirror(root);
            cout << "\nThe mirror of tree is: ";
            t.disin(root);
            break;
        case 5:
            t.search(root, key);
            break;
        case 6:
            cout << "\n***************INORDER**************" << endl;
            t.disin(root);
            break;
        case 7:
            cout << "\n***************PREORDER**************" << endl;
            t.dispre(root);
            break;
        case 8:
            cout << "\n*******************POSTORDER**************" << endl;
            t.dispost(root);
            break;
        }
        cout << "\nDo you want to continue (y/n): ";
        cin >> ans;
    } while (ans == 'y');
    return 0;
}
```

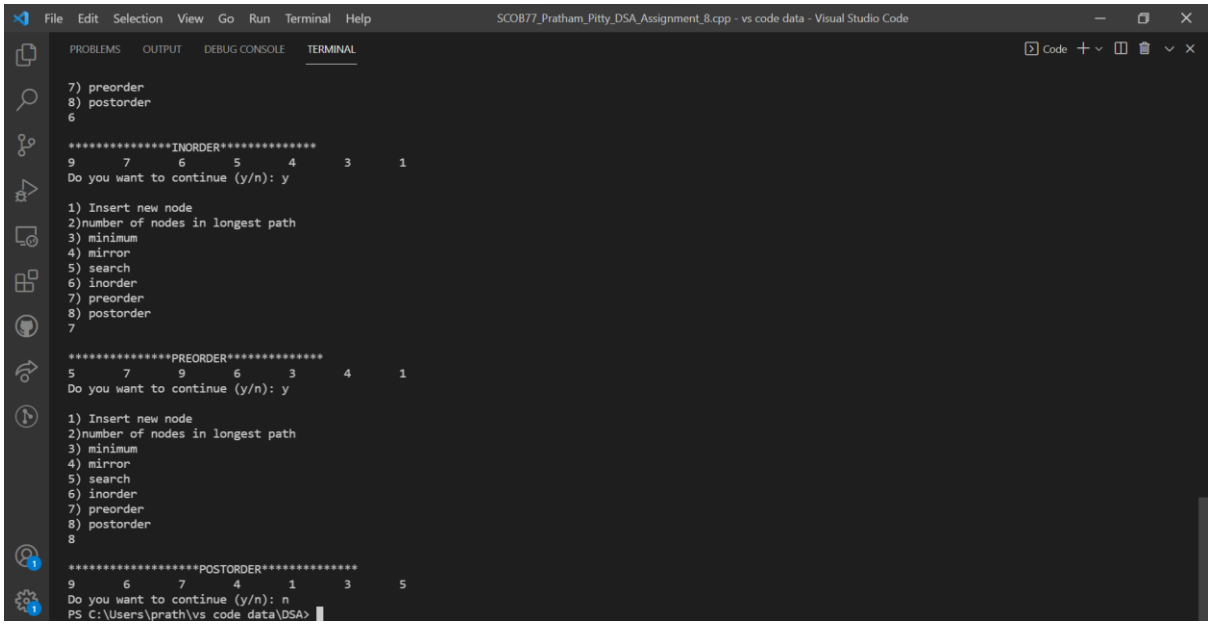
Output :-


File Edit Selection View Go Run Terminal Help
SCOB77_Pratham_Pitty_DSA_Assignment_8.cpp - vs code data - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
Warning: PowerShell detected that you might be using a screen reader and has disabled PSReadLine for compatibility purposes. If you want to re-enable it, run 'Import-Module PSReadLine'.
PS C:\Users\prath\vs code data> cd "c:\Users\prath\vs code data\DSA\" ; if ($?) { g++ SCOB77_Pratham_Pitty_DSA_Assignment_8.cpp -o SCOB77_Pratham_Pitty_DSA_Assignment_8 } ; if ($?) { .\SCOB77_Pratham_Pitty_DSA_Assignment_8 }
SCOB77_Pratham Pitty_Assignment no 8
1) Insert new node
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
1
Enter the data : 5
Do you want to insert more value :(y/n) :
y
Enter the data : 7
Do you want to insert more value :(y/n) :
y
Enter the data : 3
Do you want to insert more value :(y/n) :
y


File Edit Selection View Go Run Terminal Help
SCOB77_Pratham_Pitty_DSA_Assignment_8.cpp - vs code data - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code
Enter the data : 1
Do you want to insert more value :(y/n) :
y
Enter the data : 9
Do you want to insert more value :(y/n) :
y
Enter the data : 4
Do you want to insert more value :(y/n) :
y
Enter the data : 6
Do you want to insert more value :(y/n) :
n
The Total no.of nodes are : 7
Do you want to continue (y/n): y
1) Insert new node
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
2
Number of nodes in longest path:
3
Do you want to continue (y/n): y
1) Insert new node


File Edit Selection View Go Run Terminal Help
SCOB77_Pratham_Pitty_DSA_Assignment_8.cpp - vs code data - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
3
The min element is:
5
Do you want to continue (y/n): y
1) Insert new node
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
4
The mirror of tree is: 9 7 6 5 4 3 1
Do you want to continue (y/n): y
1) Insert new node
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
5


File Edit Selection View Go Run Terminal Help
SCOB77_Pratham_Pitty_DSA_Assignment_8.cpp - vs code data - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code
The mirror of tree is: 9 7 6 5 4 3 1
Do you want to continue (y/n): y
1) Insert new node
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
5
Enter your key :
6
KEY NOT FOUND
Do you want to continue (y/n): y
1) Insert new node
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
6
**************INORDER*************
9 7 6 5 4 3 1
Do you want to continue (y/n): y
1) Insert new node
2)number of nodes in longest path
3) minimum


File Edit Selection View Go Run Terminal Help
SCOB77_Pratham_Pitty_DSA_Assignment_8.cpp - vs code data - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Code
7) preorder
8) postorder
6
**************INORDER**************
9 7 6 5 4 3 1
Do you want to continue (y/n): y
1) Insert new node
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
7
**************PREORDER**************
5 7 9 6 3 4 1
Do you want to continue (y/n): y
1) Insert new node
2)number of nodes in longest path
3) minimum
4) mirror
5) search
6) inorder
7) preorder
8) postorder
8
******************POSTORDER**************
9 6 7 4 1 3 5
Do you want to continue (y/n): n
PS C:\Users\prath\vs code data\DSA>