AIM Binary Search Tree

QUEST Assume that there are twenty students enrolled for a spoken English class in British council. The students are roll numbered from 1 to 20. As per the registry in the entrance of the British Council, every day the entry of the students for the class is in random order. Implement a program to construct a BST (using the student Roll No) from the order in which the students are entering in to council for the class. If any student leaves the council before the end of the class, remove them from the BST. Display BST on every operation performed.

CODE

```
#include <bits/stdc++.h>
using namespace std;
class Node{
public:

   int data;

   Node *left;

   Node *right;



  public:

   Node(int x){

   data = x;

     left = NULL;

     right = NULL;

   }

};
```

```
Node *insert_BST(Node *tree, int val)

{

  if(tree == NULL){

    cout<<val <<" is inserted into BST successfully"<<endl;

    return (new Node(val));

  }

  if(tree->data >= val)

    tree->left = insert_BST(tree->left, val);

  else

    tree->right = insert_BST(tree->right, val);

  return tree;

}

bool searchInBST(Node *root , int k) {
               // Write your code here
    if(root==NULL){
        return false;
    }
    if(root->data==k){
        return true;
    }
    else if(root->data>k){
               return searchInBST(root->left ,k);
    }
    else if(root->data<k){
        return searchInBST(root->right ,k);
    }

}
Node* delete_BST(Node *tree, int val)
```

```
{

 if(tree == NULL){

   cout<<"value is not present in BST"<<endl;

   return tree;

 }

 if(tree->data > val)

   tree->left = delete_BST(tree->left, val);

 else if(tree->data < val)

   tree->right = delete_BST(tree->right, val);

 else{

   if(tree->left == NULL){

     Node *temp = tree->right;

     free(tree);

     tree= temp;

   }


   else if(tree->right == NULL){

     Node *temp = tree->left;

     tree = temp;

     free(temp);

   }
```

```
    else{
      Node *temp = tree->left;
      while(temp->right->right!=NULL){
        temp = temp->right;
      }
      tree->data = temp->right->data;
      Node *temp2 = temp->right->left;
      free(temp->right);
      temp->right = temp2;
    }
  }
  return tree;
}
void inorder(Node *tree)
{
    if (tree != NULL)
    {
        inorder(tree->left);
        cout<<tree->data<<" ";
        inorder(tree->right);
    }
```

```
}
void printLevelATNewLine(Node *root) {
    if (root == NULL) {
        return;
    }
    queue<Node *> q;
    q.push(root);
    q.push(NULL);
    while (!q.empty()) {
        Node *first = q.front();
        q.pop();
        if (first == NULL) {
            if (q.empty()) {
                break;
            }
            cout << endl;
            q.push(NULL);
            continue;
        }
        cout << first->data << " ";
        if (first->left != NULL) {
            q.push(first->left);
        }
        if (first->right != NULL) {
            q.push(first->right);
        }
    }
}

int main(){
    cout<<"BRITISH COUNCIL CLASS"<<endl;
    // cout<<"Root of the BST i.e. the first child ";
    // int x;
    // cin>>x;
    // cout<<x<<endl;
    Node *root = NULL;
    //takeInput(root);

    int z;
    while(z!=6){
    cout<<"*****************"<<endl;
```

```
cout<<"Enter 1: To Insert student"<<endl;
cout<<"Enter 2: To Search for any student"<<endl;
cout<<"Enter 3: if any student exits"<<endl;
cout<<"Enter 4: To print the tree in order"<<endl;
cout<<"Enter 5: To print level wise"<<endl;
cout<<"Enter 6: To Exit"<<endl;
cout<<"*****************"<<endl;

cin >> z;
switch(z){
  case 1:
  cout<<"Enter the student roll no. as they enter the class "<<endl;
  int id;
  cin>>id;
  root=insert_BST(root, id);
  inorder(root);
  break;
  case 2:
  int k;
  cout<<"Enter the student Roll No. for search "<<endl;
  cin>>k;
  if(searchInBST(root,k)==false){
    cout<<"Student is NOT PRESENT in the class"<<endl;
  }
  else{
    cout<<"Student is PRESENT in the class"<<endl;
  }
  break;
  case 3:
  int m;
  cout<<"Enter the student's Roll No. who exited "<<endl;
  cin>>m;
  delete_BST(root, m);
  //printLevelATNewLine(root);
  break;
  case 4:
  inorder(root);
  break;
  case 5:
  printLevelATNewLine(root);
  break;
  case 6:
      z=6;
      break;
   }

   }
   // cout << ((searchInBST(root, k)) ? "true" : "false");
   cout<<"*******EXIT********"<<endl;
   delete root;

}
```

PSEUDO CODE

INSERTION

```
Step1 If tree is empty (no root), create a node holding key k as root; done.

Step2. If the val if less than the root we will traverse the left side.

Step3. If the val if greater than the root we will traverse the right side.
Step4. If k< CurrNode.key ... /* key must go in left subtree */
If CurrNode.left == NULL, create a node holding k as left child of CurrNode; done.
else set CurrNode = CurrNode.left, and go to 3.

Step5. else ... /* key must go in right subtree */
If CurrNode.right == NULL, create a node holding k as left child of CurrNode; done.
else set CurrNode = CurrNode.right, and go to 3.
```

SEARCH

```
Step1 Base CASE-If root is empty return empty;

Step2 If root is equal to the given value return true;

Step3 if root is greater than the val traverse left

Step4 if root is less than the val traverse right
```

DELETE

Step1 Base CASE-If tree==NULL,print “not present” and return tree;

Step2 If root is equal to the given value, delete the root node and connect everything;

Step3 if root is greater than the val traverse left

Step4 if root is less than the val traverse right

INORDER

Step1 check if tree not equal to null;

Step2 start printing from the left most node then the root then the right;

PRINT LEVEL WISE

Step1 if root is equal to NULL return;

Step2 define a queue “q”;

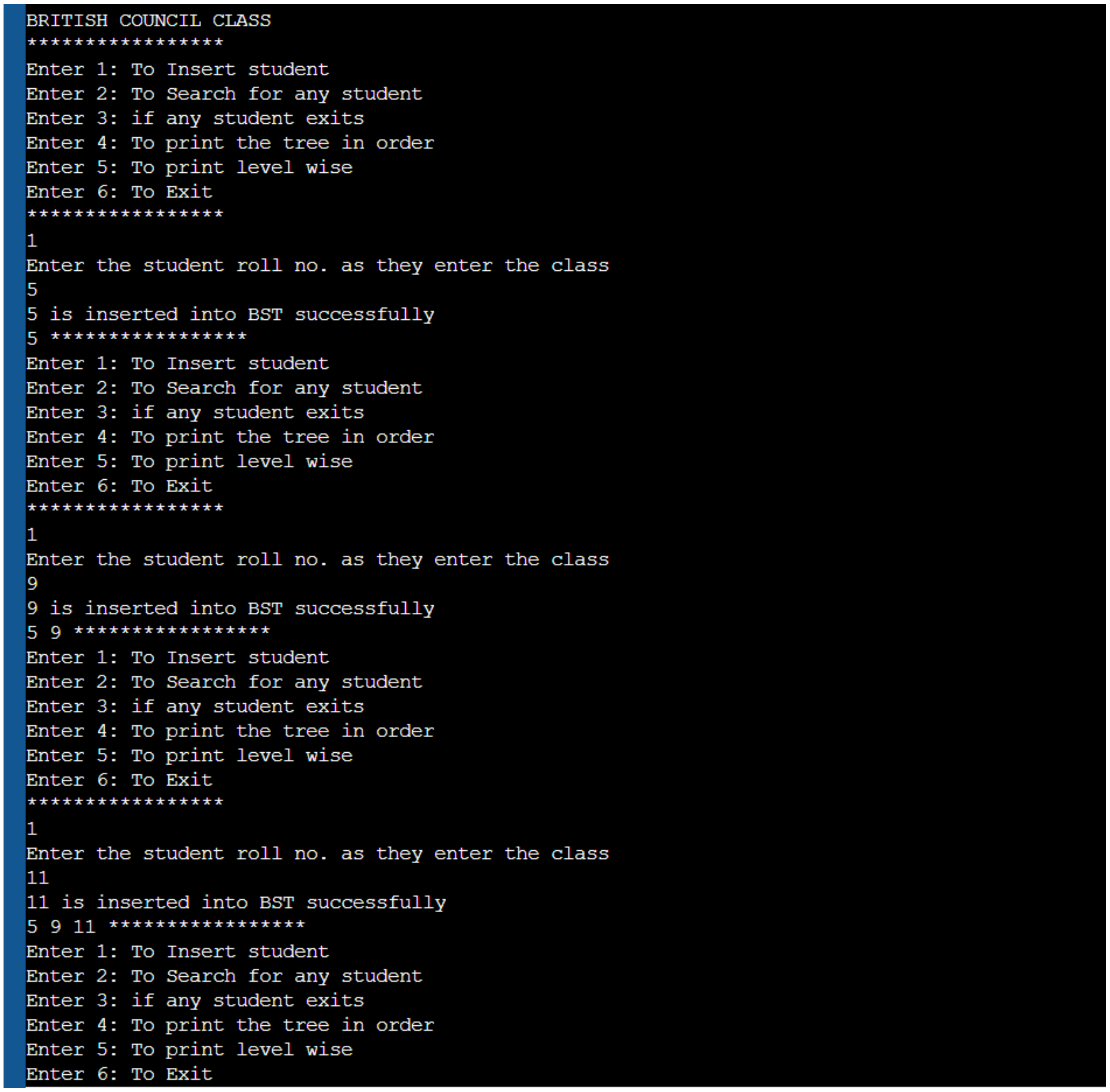
Step3 push the first element;

Step4 start the loop till queue is empty

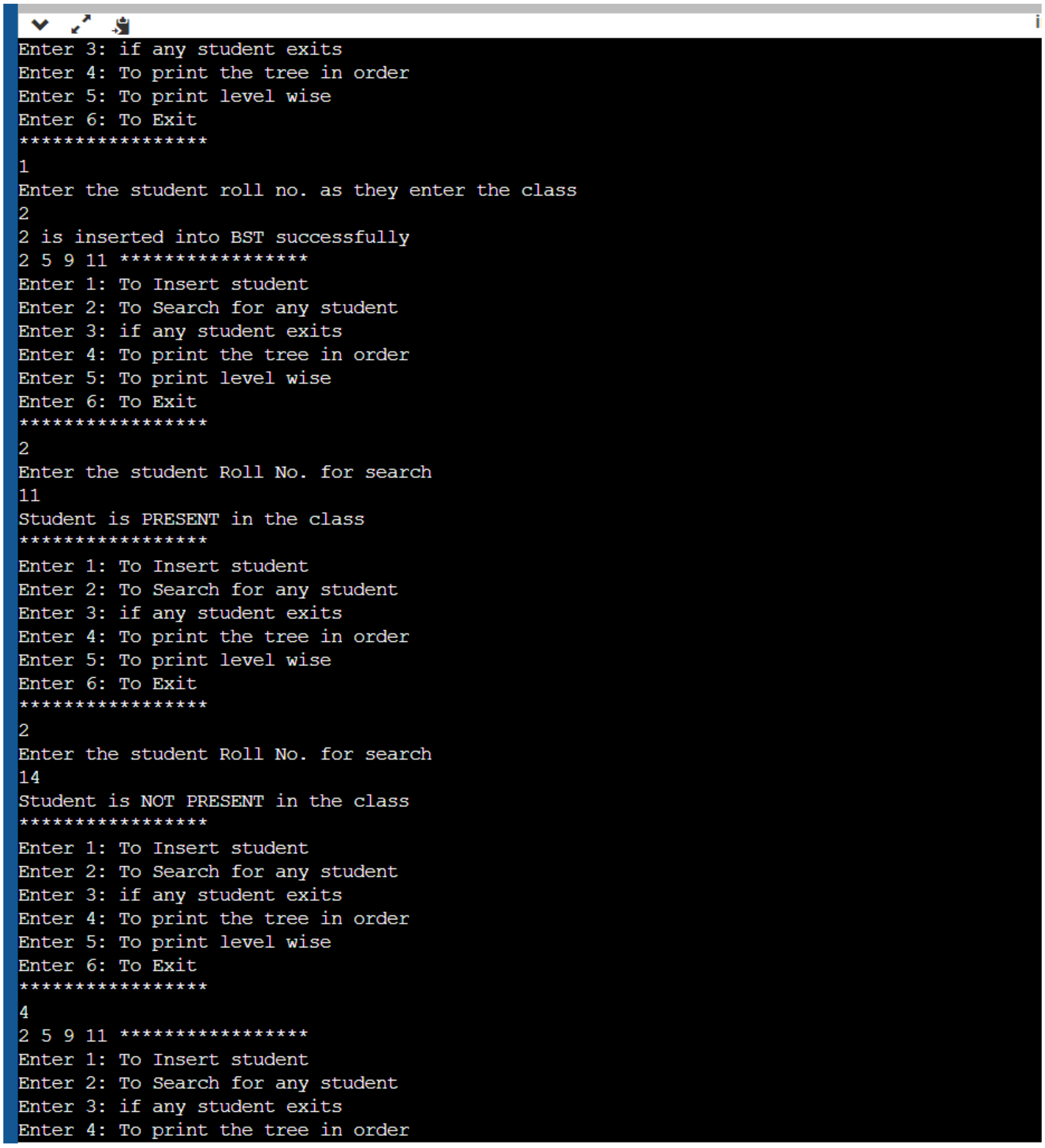
Step5 store the front node in “first” and pop

Step 6 END

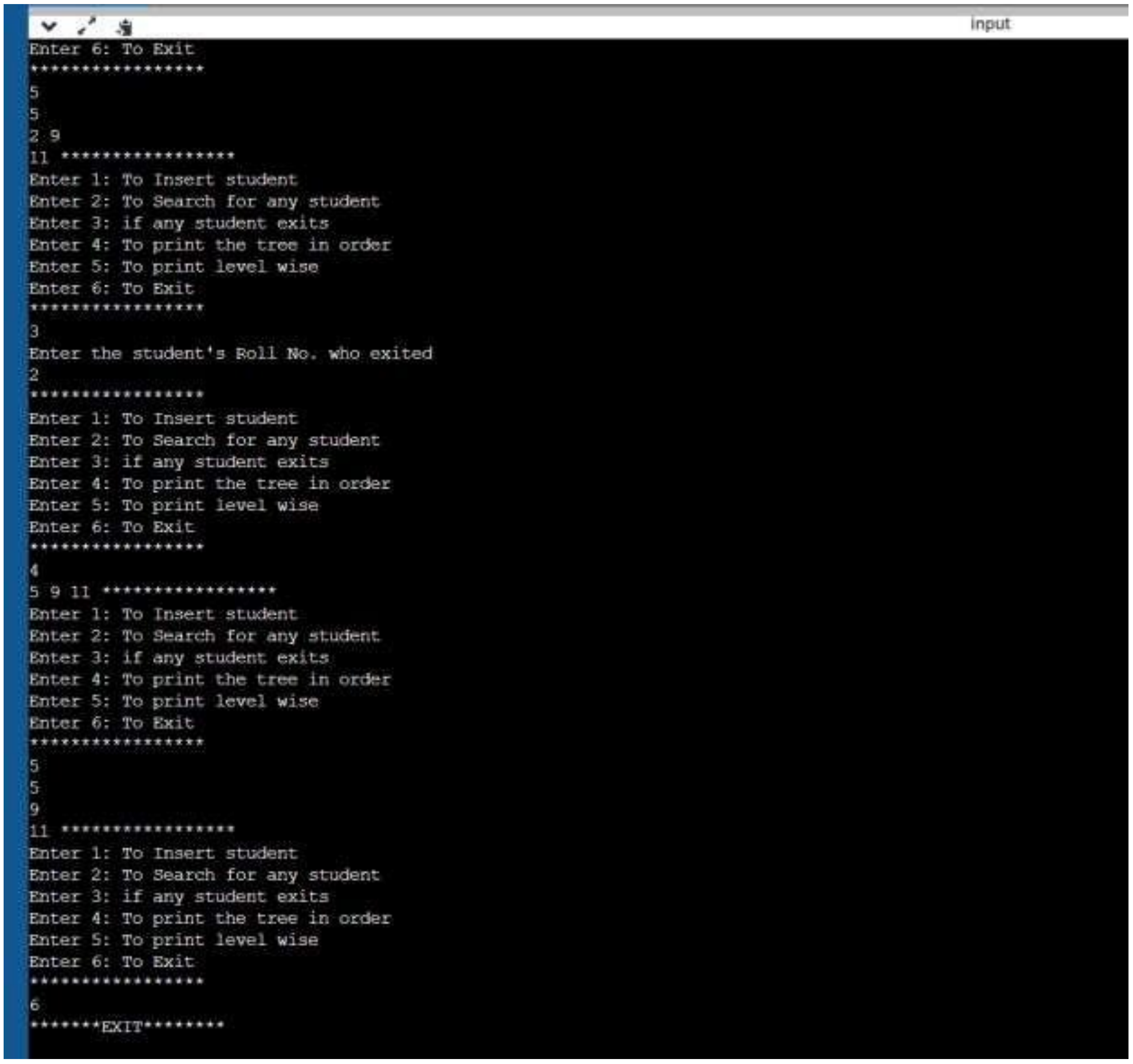
OUTPUT



```
BRITISH COUNCIL CLASS
*****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
1
Enter the student roll no. as they enter the class
5
5 is inserted into BST successfully
5 *****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
1
Enter the student roll no. as they enter the class
9
9 is inserted into BST successfully
5 9 *****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
1
Enter the student roll no. as they enter the class
11
11 is inserted into BST successfully
5 9 11 *****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
```



```
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
1
Enter the student roll no. as they enter the class
2
2 is inserted into BST successfully
2 5 9 11 *****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
2
Enter the student Roll No. for search
11
Student is PRESENT in the class
*****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
2
Enter the student Roll No. for search
14
Student is NOT PRESENT in the class
*****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
4
2 5 9 11 *****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
```



```
Enter 6: To Exit
*****************
5
5
2 9
11 *****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
3
Enter the student's Roll No. who exited
2
*****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
4
5 9 11 *****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
5
5
9
11 *****************
Enter 1: To Insert student
Enter 2: To Search for any student
Enter 3: if any student exits
Enter 4: To print the tree in order
Enter 5: To print level wise
Enter 6: To Exit
*****************
6
*******EXIT********
```