DSA LAB – 10

Name: Etcherla Sai Manoj **Mis. No:** 112015044 **Branch:** CSE

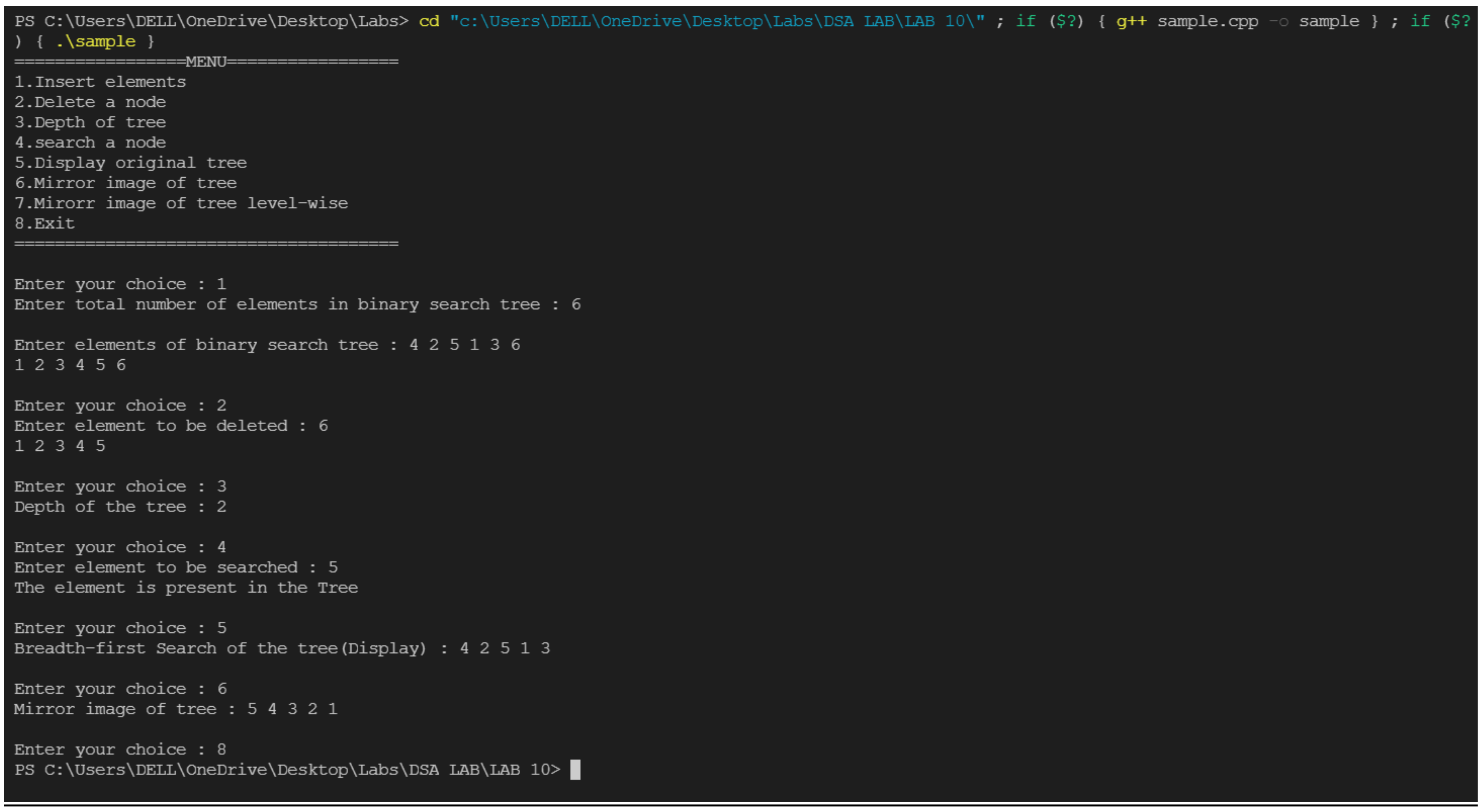
Question 1:

Code:

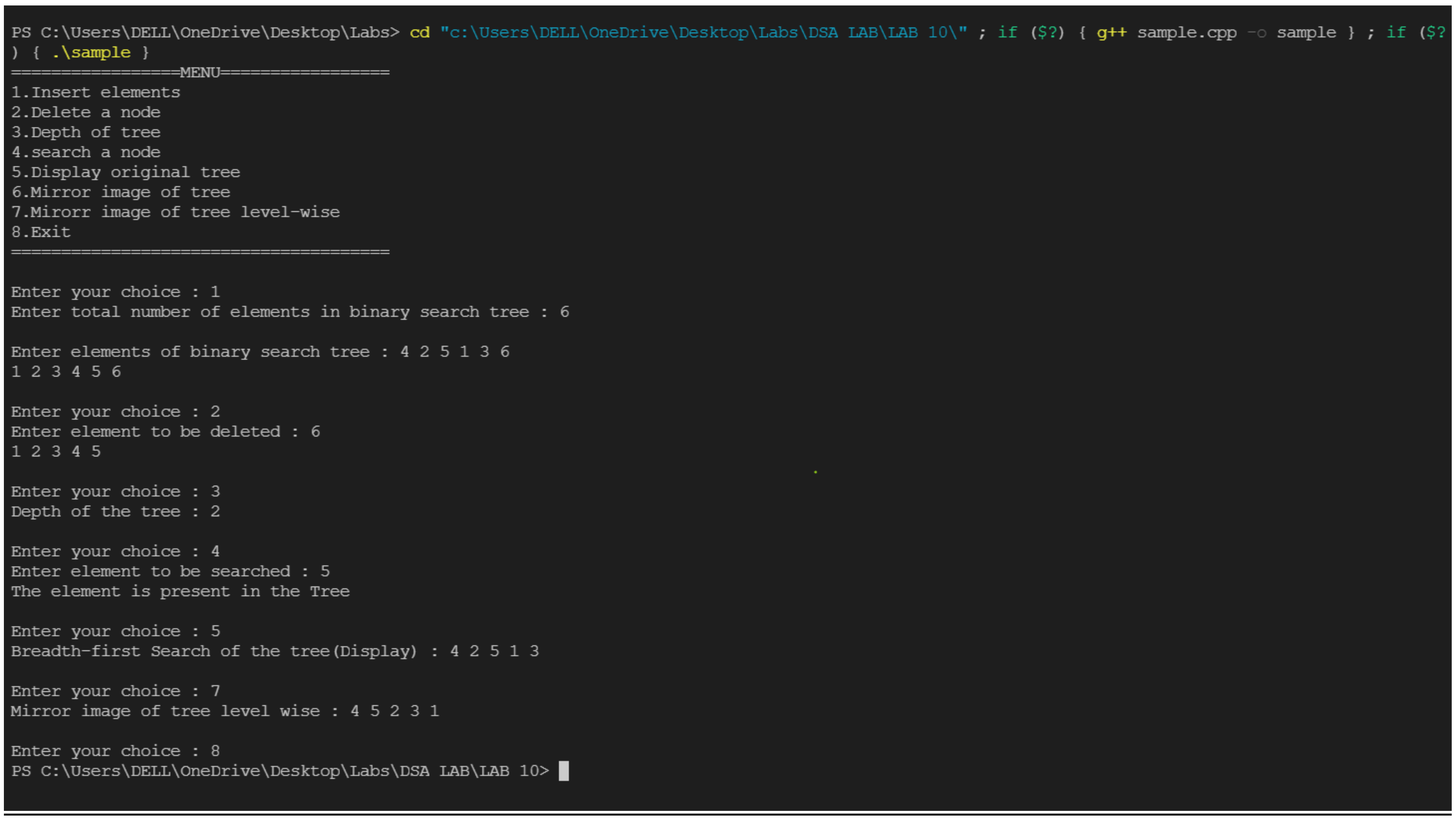
```
#include<bits/stdc++.h>
using namespace std;
struct Node{
   int data;
   Node* left, *right;
};

Node* create_new_node(int data){
   Node* temp = new Node();
   temp->data = data;
   temp->left = temp->right = NULL;
   return temp;
}

Node* insert_node(Node* ptr, int data){
   if(ptr == NULL){
      ptr = create_new_node(data);
      return ptr;
   }
   else if(ptr->data >= data) ptr->left = insert_node(ptr->left,data);
   else ptr->right = insert_node(ptr->right,data);
   return ptr;
}

void search(Node* ptr, int data){
   if(ptr == NULL){
      cout << "The element is not present in the Tree\n";
      return;
   }
   if(ptr->data == data){
      cout << "The element is present in the Tree\n";
      return;
   }
   if(ptr->data > data){
      search(ptr->left, data);
   }
   else{
      search(ptr->right, data);
   }
}

void depthfirst_display(Node* ptr){
   if(ptr == NULL) return;
   depthfirst_display(ptr->left);
   cout << ptr->data <<" ";
   depthfirst_display(ptr->right);
}

void breadthfirst_display(Node* ptr){
   if(ptr == NULL){
      cout << "Tree is Empty\n";
      return;
   }
   queue<Node*> q1;
   q1.push(ptr);
   while(!q1.empty()){
      Node* temp = q1.front();
      cout << temp->data << " ";
      if(temp->left != NULL){
            q1.push(temp->left);
        }
        if(temp->right != NULL){
            q1.push(temp->right);
        }
        q1.pop();
    }
}

int minimum(Node* ptr){
    if(ptr == NULL) return -1;
    while(ptr->left!=NULL){
        ptr = ptr->left;
    }
    return ptr->data;
}

Node* deletenode(Node* ptr,int data){
    if(ptr == NULL) return ptr;
    if(ptr->data > data){
        ptr->left = deletenode(ptr->left, data);
    }
    else if(ptr->data < data){
        ptr->right = deletenode(ptr->right, data);
    }
    else{
        if(ptr->left == NULL and ptr->right == NULL){
            delete ptr;
            ptr = NULL;
            return ptr;
        }
        else if(ptr->right == NULL){
            Node* temp = ptr;
            ptr = ptr->left;
            delete temp;
            return ptr;
        }
        else if(ptr->left == NULL){
            Node* temp = ptr;
            ptr = ptr->right;
            delete temp;
            return ptr;
        }
        else{
            int right_minimum = minimum(ptr->right);
            ptr->data = right_minimum;
            ptr->right = deletenode(ptr->right, right_minimum);
        }
    }
    return ptr;
}

int depth(Node* ptr){
    if(ptr == NULL) return -1;
    return max(depth(ptr->right), depth(ptr->left)) + 1;
}

void mirror(Node* ptr){
    if (ptr == NULL) return;
    else{
        struct Node* temp;
        mirror(ptr->left);
        mirror(ptr->right);
        temp = ptr->left;
        ptr->left = ptr->right;
        ptr->right = temp;
    }
}
```

```
int main(){
   Node* head = NULL;
   int test_node, insert, choice;
   cout << "=================MENU=================\n";
   cout << "1.Insert elements\n";
   cout << "2.Delete a node\n";
   cout << "3.Depth of tree\n";
   cout << "4.search a node\n";
   cout << "5.Display original tree\n";
   cout << "6.Mirror image of tree\n";
   cout << "7.Mirorr image of tree level-wise\n";
   cout << "8.Exit\n";
   cout << "======================================\n";

   while(1){
      cout << "\nEnter your choice : ";
      cin >> choice;
      switch(choice){
         case 1:
            cout << "Enter total number of elements in binary search tree : ";
            cin >> test_node;
            cout << "\nEnter elements of binary search tree : ";
            //inserting nodes to tree
            for(int j=0; j < test_node; j++){
               cin >> insert;
               head = insert_node(head, insert);
            }
            depthfirst_display(head);
            cout << "\n";
            break;
         case 2:
            cout<<"Enter element to be deleted : ";
            cin >> test_node;
            head = deletenode(head, test_node);
            depthfirst_display(head);
            cout << "\n";
            break;
         case 3:
            cout << "Depth of the tree : " << depth(head);
            cout << "\n";
            break;
         case 4:
            cout << "Enter element to be searched : ";
            cin >> test_node;
            search(head, test_node);
            break;
         case 5:
            cout<<"Breadth-first Search of the tree(Display) : ";
            breadthfirst_display(head);
            cout << "\n";
            break;
         case 6:
            cout << "Mirror image of tree : ";
            mirror(head);
            depthfirst_display(head);
            cout << "\n";
            break;
         case 7:
            cout << "Mirror image of tree level wise : ";
            mirror(head);
            breadthfirst_display(head);
            cout << "\n";
            break;
         case 8:
            return 0;
         default:
            cout << "Enter a valid choice!!!\n";
            break;
        }
    }
    return 0;
}
```

Input & Output:



```
PS C:\Users\DELL\OneDrive\Desktop\Labs> cd "c:\Users\DELL\OneDrive\Desktop\Labs\DSA LAB\LAB 10\" ; if ($?) { g++ sample.cpp -o sample } ; if ($?
) { .\sample }
=================MENU=================
1.Insert elements
2.Delete a node
3.Depth of tree
4.search a node
5.Display original tree
6.Mirror image of tree
7.Mirorr image of tree level-wise
8.Exit
======================================

Enter your choice : 1
Enter total number of elements in binary search tree : 6

Enter elements of binary search tree : 4 2 5 1 3 6
1 2 3 4 5 6

Enter your choice : 2
Enter element to be deleted : 6
1 2 3 4 5

Enter your choice : 3
Depth of the tree : 2

Enter your choice : 4
Enter element to be searched : 5
The element is present in the Tree

Enter your choice : 5
Breadth-first Search of the tree(Display) : 4 2 5 1 3

Enter your choice : 6
Mirror image of tree : 5 4 3 2 1

Enter your choice : 8
PS C:\Users\DELL\OneDrive\Desktop\Labs\DSA LAB\LAB 10>
```



```
PS C:\Users\DELL\OneDrive\Desktop\Labs> cd "c:\Users\DELL\OneDrive\Desktop\Labs\DSA LAB\LAB 10\" ; if ($?) { g++ sample.cpp -o sample } ; if ($?
) { .\sample }
=================MENU=================
1.Insert elements
2.Delete a node
3.Depth of tree
4.search a node
5.Display original tree
6.Mirror image of tree
7.Mirorr image of tree level-wise
8.Exit
======================================

Enter your choice : 1
Enter total number of elements in binary search tree : 6

Enter elements of binary search tree : 4 2 5 1 3 6
1 2 3 4 5 6

Enter your choice : 2
Enter element to be deleted : 6
1 2 3 4 5

Enter your choice : 3
Depth of the tree : 2

Enter your choice : 4
Enter element to be searched : 5
The element is present in the Tree

Enter your choice : 5
Breadth-first Search of the tree(Display) : 4 2 5 1 3

Enter your choice : 7
Mirror image of tree level wise : 4 5 2 3 1

Enter your choice : 8
PS C:\Users\DELL\OneDrive\Desktop\Labs\DSA LAB\LAB 10>
```