```
#include <iostream>
using namespace std;
struct node
{
   char label[20];
   int ch_count;
   struct node *child[10];
}*root;
class BST
{
   public:
   void create();
   void display(node *r1);
   BST()
   {
      root=NULL;
   }
};
void BST::create()
{
   int i ,j, k, tchapters, tbook;
   root=new node;
   cout<<"Enter the name of the book "<<endl;
   cin>>root->label;
   cout<<"Enter the number of chapters the book contains ";
   cin>>tchapters;
   root->ch_count=tchapters;
   for(i=0; i<tchapters; i++)
   {
      root->child[i]=new node;
      cout<<"Enter the name of the chapter ";
      cin>>root->child[i]->label;

      cout<<"Enter the number of sections in this chapter ";
      cin>>root->child[i]->ch_count;
```

```
      for(j=0; j<root->child[i]->ch_count; j++)
      {
         root->child[i]->child[j]=new node;
         cout<<"Enter section head ";
         cin>>root->child[i]->child[j]->label;
         cout<<"Enter the number of sub-sections in this chapter ";
         cin>>root->child[i]->child[j]->ch_count;
         for(k=0; k<root->child[i]->child[j]->ch_count; k++)
         {
            root->child[i]->child[j]->child[k]=new node;
            cout<<"Enter sub-section head ";
            cin>>root->child[i]->child[j]->child[k]->label;
         }
      }
   }
}
void BST::display(node *r1)
{
   int i,j,k,tchapters;
   if (r1!=NULL)
   {
      cout<<"\n  -----Book Hierarchy-----";
      cout<<"\n BOOK TITLE: "<<r1->label;
      tchapters=r1->ch_count;
      for(i=0; i<tchapters; i++)
      {
         cout<<"\n----------------"<<endl;
         cout<<"\n CHAPTER: "<<i+1<<". ";
         cout<<r1->child[i]->label<<endl;

         for(j=0; j<r1->child[i]->ch_count; j++)
         {
            cout<<"\n SECTION: ";
            cout<<r1->child[i]->child[j]->label<<endl;
            for(k=0; k<r1->child[i]->child[j]->ch_count; k++)
            {
```

```
            cout<<"\n SUB-SECTION: ";
            cout<<r1->child[i]->child[j]->child[k]->label<<endl;
          }
        }
      }
    }
}
int main()
{
   int choice  ;
   BST bst;
   while(1)
   {
      cout<<"\n\n----------"<<endl;
      cout<<"Book tree creation"<<endl;
      cout<<"----------"<<endl;
      cout<<"1.Create "<<endl;
      cout<<"2.Display "<<endl;
      cout<<"3.Quit "<<endl;
      cout<<"Enter your choice: ";
      cin>>choice;
      switch(choice )
      {
         case 1:
         bst.create();
         case 2:
         bst.display(root);
         break;
         case 3:
         exit(1);
         default:
         cout<<"Wrong choice "<<endl;
      }
   }
   return 0;
}
```

```
--------------
Book tree creation
---------------
1.Create
2.Display
3.Quit
Enter your choice: 1
Enter the name of the book
Java
Enter the number of chapters the book contains 2
Enter the name of the chapter Hashing
Enter the number of sections in this chapter 2
Enter section head Synchronized
Enter the number of sub-sections in this chapter 1
Enter sub-section head HashTable
Enter section head Asynchronized
Enter the number of sub-sections in this chapter 1
Enter sub-section head HashMap
Enter the name of the chapter Trees
Enter the number of sections in this chapter 2
Enter section head BT
Enter the number of sub-sections in this chapter 0
Enter section head BST
Enter the number of sub-sections in this chapter 0
```

```
 -----Book Hierarchy-----
BOOK TITLE: Java
---------------

CHAPTER: 1. Hashing

SECTION: Synchronized

SUB-SECTION: HashTable

SECTION: Asynchronized

SUB-SECTION: HashMap

---------------

CHAPTER: 2. Trees

SECTION: BT

SECTION: BST
```

```
#include<iostream>
#include<fstream>
#include<cstdio>
using namespace std;
class student
{
   int admno;
   char name[50];
   char addr[50];
   char divv[50];
   public:
   void setdata()
   {
      cout<<"\nEnter roll number of the student ";
      cin>>admno;
      cout<<"\nEnter name of the student ";
      cin>>name;
      cout<<"\nEnter the division of the student ";
      cin>>divv;
      cout<<"\nEnter the address of the student ";
      cin>>addr;
   }
   void showdata ()
   {
      cout<< "\n*Student Roll No: "<<admno<<endl;
      cout<<"*Student Name: "<<name<<endl;
      cout<<"*Student Division: "<<divv<<endl;
      cout<<"*Student Address: "<<addr<<endl;
   }
   int retadmno()
   {
      return admno;
   }
};
```

```
void write_record()
{
   ofstream outfile;
   outfile.open("student.dat",ios::binary | ios::app);
   student obj;
   obj.setdata();
   outfile.write((char*)&obj, sizeof(obj));
   outfile.close();
}
void display()
{
   ifstream infile;
   infile.open("student.dat", ios::binary );
   student obj;
   while(infile.read((char*)&obj, sizeof(obj)))
   {
      obj.showdata();
   }
   infile.close();
}
void search(int n)
{
   ifstream infile;
   infile.open("student.dat", ios::binary);
   int flag=0;
   student obj;
   while(infile.read((char*)&obj, sizeof(obj)))
   {
      if(obj.retadmno()==n)
      {
         obj.showdata();
         flag=1;
         break;
      }
   }
```

```
    if (flag==0)
    {
        cout<<"\nRecord not found "<<endl;
    }
    infile.close();
}
void delete_record(int n)
{
    student obj;
    ifstream infile;
    infile.open("student.dat", ios::binary);
    ofstream outfile;
    outfile.open("temp.dat",ios::out | ios::binary);
    while(infile.read((char*)&obj, sizeof(obj)))
    {
        if (obj.retadmno()!=n)
        {
            outfile.write((char*)&obj, sizeof(obj));
        }
    }
    infile.close(); outfile.close();

    remove("student.dat");
    rename("temp.dat","student.dat");
}

int main()
{
    int ch;
    do {
        cout<<"\n\n\n****File Operations*** "<<endl;
        cout<<"\n1.Write \n2.Display \n3.Search \n4.Delete "<<endl;
        cout<<"Enter your choice "; cin>>ch;
        switch(ch)
        {
            case 1:
```

```
        int n;
        cout<<"\nEnter number of records ";cin>>n;
        for(int i=0; i<n; i++)
        {
           write_record();
        }
        break;

        case 2:
        cout<<"\nList of records "<<endl;
        display();
        break;

        case 3:
        int s;
        cout<<"\nEnter the student's roll number you want to search for :";cin>>s;
        search(s);
        break;

        case 4:
        cout<<"\nEnter the number to be deleted ";
        int d;
        cin>>d;
        delete_record(d);
        break;

        default:
        cout<<"Wrong choice ";
        break;
      }
   }while(ch!=5);
   return 0;
}
```

```
1.Write
2.Display
3.Search
4.Delete
Enter your choice 1

Enter number of records 2

Enter roll number of the student 7

Enter name of the student Siddhesh

Enter the division of the student A

Enter the address of the student Chinchwad

Enter roll number of the student 55

Enter name of the student Rushikesh

Enter the division of the student B

Enter the address of the student Bophkel
```

```
****File Operations***

1.Write
2.Display
3.Search
4.Delete
Enter your choice 2

List of records

*Student Roll No: 7
*Student Name: Siddhesh
*Student Division: A
*Student Address: Chinchwad

*Student Roll No: 55
*Student Name: Rushikesh
*Student Division: B
*Student Address: Bophkel
```

```
1.Write
2.Display
3.Search
4.Delete
Enter your choice 3

Enter the student's roll number you want to search for :55

*Student Roll No: 55
*Student Name: Rushikesh
*Student Division: B
*Student Address: Bophkel



****File Operations***

1.Write
2.Display
3.Search
4.Delete
Enter your choice 4

Enter the number to be deleted 7
```

```
****File Operations***

1.Write
2.Display
3.Search
4.Delete
Enter your choice 2

List of records

*Student Roll No: 55
*Student Name: Rushikesh
*Student Division: B
*Student Address: Bophkel
```

```
#include<iostream>
using namespace std;
class Node
{
public:
   int key;
   Node *ln, *rn;
};
class Tree
{
public:
   Node* root;
 Node* createTree(int key)
   {
      root = new Node();
      root->key = key;
      root->ln = NULL;
      root->rn = NULL;
            return root;
   }
   void insertNode(int key, Node* root)
   {
      Node* node = new Node();
      node->key = key;
      if (root->key >key)      //COMPARISION
      {
         if (root->ln == NULL)
         {
            root->ln = node;
         }
         else
            insertNode(key, root->ln);
      }
      else if (root->key < key)
```

```
    {
       if (root->rn == NULL)
       {
          root->rn = node;
       }
       else
          insertNode(key, root->rn);
    }
         else
         cout<<"No duplicate keys are allowed"<<endl;
}
void searchNode(int searchkey, Node* root)
{
    if(root == NULL)
                    cout<<"No tree present";

                    if(root->key==searchkey)
                    {
                              cout<<"Key found !!!"<<endl;
                    }

                    else if (root->key > searchkey)
         {
         if (root->ln == NULL)
         {
           cout<<"Key is not present in the tree"<<endl;
         }
         else
          searchNode(searchkey, root->ln);
         }
         else if (root->key < searchkey)
                        {
                         if (root->rn == NULL)
          {
```

```
            cout<<"Key is not present in the tree"<<endl;
          }


           else
           searchNode(searchkey, root->rn);
         }
}
void displayInorder(Node* root)
{
   if (root != NULL)
   {
      displayInorder(root->ln);
      cout << root->key << endl;
      displayInorder(root->rn);
   }
}
void displaymin(Node* root)
{
   while (root->ln != NULL)
   {
      root = root->ln;
   }
cout<<"Minimum number is " << root->key <<endl;
}
void displaymax(Node* root){
   while(root->rn != NULL){
      root = root->rn;
   }
   cout<<"Maximum number is " << root->key <<endl;
}
int longestPath(Node* root)
{
   if(root==NULL)
      return 0;
   int Lctr = longestPath(root->ln);
   int Rctr = longestPath(root->rn);
```

```
        if(Lctr>Rctr)
            return (Lctr+1);
        else return (Rctr+1);
    }

    Node* swapNodes(Node* root)
    {
        Node* temp;
        if(root==NULL)
            return NULL;
        temp = root->ln;       //SWAPPING
        root->ln=root->rn;
        root->rn=temp;
        swapNodes(root->ln);
        swapNodes(root->rn);
    }
};
int main()
{
    int choice, order, flag = 0;
    int key, searchKey;
    Tree t1;
    Node* root;
    do
    {
        cout<<"   MENU "<<endl;
        cout<<"1. Insert Node "<<endl;
        cout<<"2. Display Inorder of the Tree"<<endl;
        cout<<"3. Display Min"<<endl;
        cout<<"4. Display Max"<<endl;
        cout<<"5. Swap left and right subtrees"<<endl;
        cout<<"6. Search in tree"<<endl;
        cout<<"7. Number of nodes in the longest path"<<endl;
        cout<<"8. EXIT "<<endl;
        cout<<"Enter choice: ";
        cin >> choice;
```

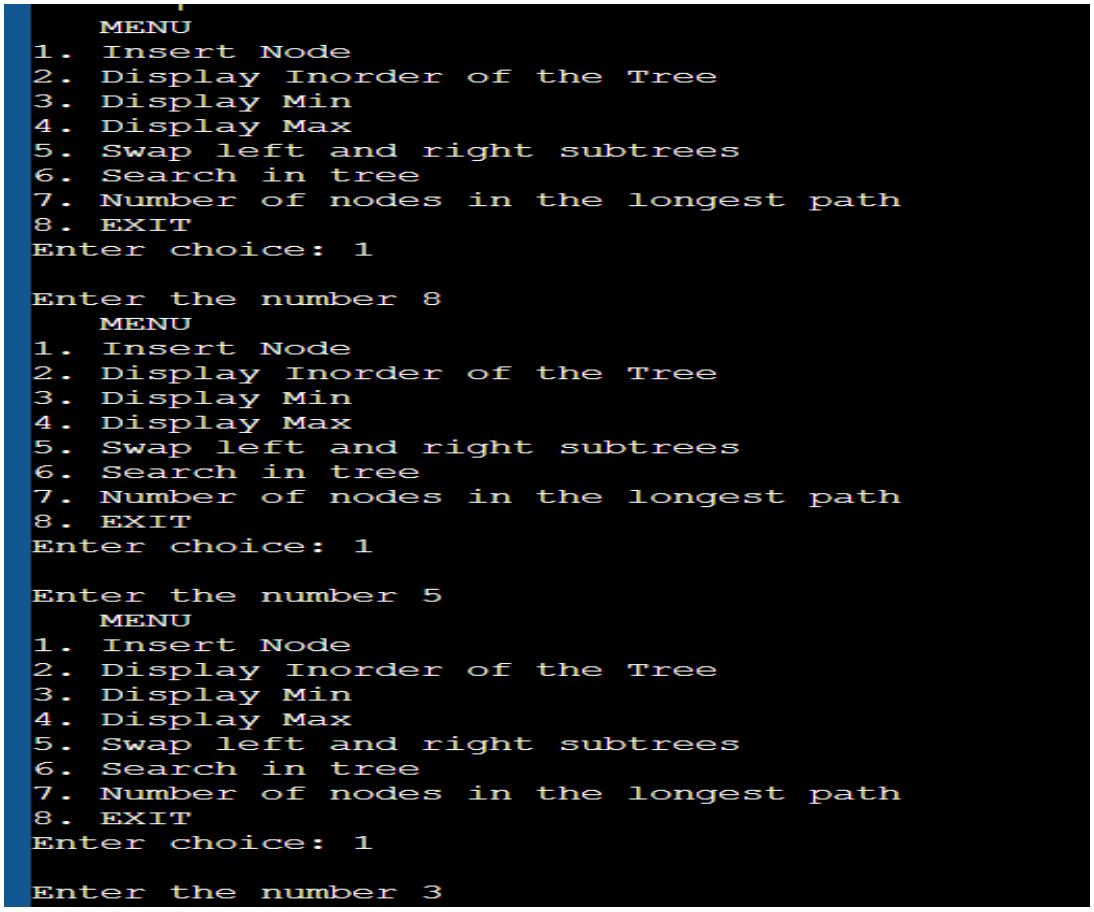
```
switch (choice)
{
case 1:
   cout << "\nEnter the number ";

   cin >> key;
   if (flag == 0)
   {
      root = t1.createTree(key);
      flag = 1;
   }
   else
   {
      t1.insertNode(key, root);
   }
   break;

case 2:
      t1.displayInorder(root);
      break;
case 3:
      t1.displaymin(root);
      break;
case 4:
      t1.displaymax(root);
      break;
case 5:
      t1.swapNodes(root);
      cout<<"Swapped! The new list is : ";
      t1.displayInorder(root);
      break;
case 6:
      cout << "\nEnter the key you want to search: ";
      cin >> searchKey;
      t1.searchNode(searchKey,root);
      break;
```

```
    case 7:
        cout<<"Number of nodes in the longest path is : "<<t1.longestPath(root);
        break;
    case 8:
      exit(0);
    }
  }
  while (choice != 8);
  return 0;
}
```

OUTPUT



```
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 1

Enter the number 6
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 1

Enter the number 12
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 1
```

```
Enter the number 10
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 1

Enter the number 14
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 2
3
5
6
8
10
12
14
```

```
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 3
Minimum number is 3
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 6

Enter the key you want to search: 12
Key found !!!
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 7
Number of nodes in the longest path is :3    MENU
```

```
Number of nodes in the longest path is :3   MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 5
Swapped! The new list is : 14
12
10
8
6
5
3
   MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 8
```

```
#include<iostream>
using namespace std;
class Node
{
public:
   int key;
   Node *ln, *rn;
};
class Tree
{
public:
   Node* root;
 Node* createTree(int key)
   {
      root = new Node();
      root->key = key;
      root->ln = NULL;
      root->rn = NULL;
            return root;
   }
   void insertNode(int key, Node* root)
   {
      Node* node = new Node();
      node->key = key;
      if (root->key >key)      //COMPARISION
      {
         if (root->ln == NULL)
         {
            root->ln = node;
         }
         else
            insertNode(key, root->ln);
      }
      else if (root->key < key)
```

```
    {
       if (root->rn == NULL)
       {
          root->rn = node;
       }
       else
          insertNode(key, root->rn);
    }
         else
         cout<<"No duplicate keys are allowed"<<endl;
}
void searchNode(int searchkey, Node* root)
{
    if(root == NULL)
                    cout<<"No tree present";

                    if(root->key==searchkey)
                    {
                              cout<<"Key found !!!"<<endl;
                    }

                    else if (root->key > searchkey)
         {
         if (root->ln == NULL)
         {
           cout<<"Key is not present in the tree"<<endl;
         }
         else
          searchNode(searchkey, root->ln);
         }
         else if (root->key < searchkey)
                        {
                         if (root->rn == NULL)
          {
```

```
            cout<<"Key is not present in the tree"<<endl;
          }


           else
           searchNode(searchkey, root->rn);
         }
}
void displayInorder(Node* root)
{
   if (root != NULL)
   {
      displayInorder(root->ln);
      cout << root->key << endl;
      displayInorder(root->rn);
   }
}
void displaymin(Node* root)
{
   while (root->ln != NULL)
   {
      root = root->ln;
   }
cout<<"Minimum number is " << root->key <<endl;
}
void displaymax(Node* root){
   while(root->rn != NULL){
      root = root->rn;
   }
   cout<<"Maximum number is " << root->key <<endl;
}
int longestPath(Node* root)
{
   if(root==NULL)
      return 0;
   int Lctr = longestPath(root->ln);
   int Rctr = longestPath(root->rn);
```

```
        if(Lctr>Rctr)
            return (Lctr+1);
        else return (Rctr+1);
    }

    Node* swapNodes(Node* root)
    {
        Node* temp;
        if(root==NULL)
            return NULL;
        temp = root->ln;       //SWAPPING
        root->ln=root->rn;
        root->rn=temp;
        swapNodes(root->ln);
        swapNodes(root->rn);
    }
};
int main()
{
    int choice, order, flag = 0;
    int key, searchKey;
    Tree t1;
    Node* root;
    do
    {
        cout<<"   MENU "<<endl;
        cout<<"1. Insert Node "<<endl;
        cout<<"2. Display Inorder of the Tree"<<endl;
        cout<<"3. Display Min"<<endl;
        cout<<"4. Display Max"<<endl;
        cout<<"5. Swap left and right subtrees"<<endl;
        cout<<"6. Search in tree"<<endl;
        cout<<"7. Number of nodes in the longest path"<<endl;
        cout<<"8. EXIT "<<endl;
        cout<<"Enter choice: ";
        cin >> choice;
```

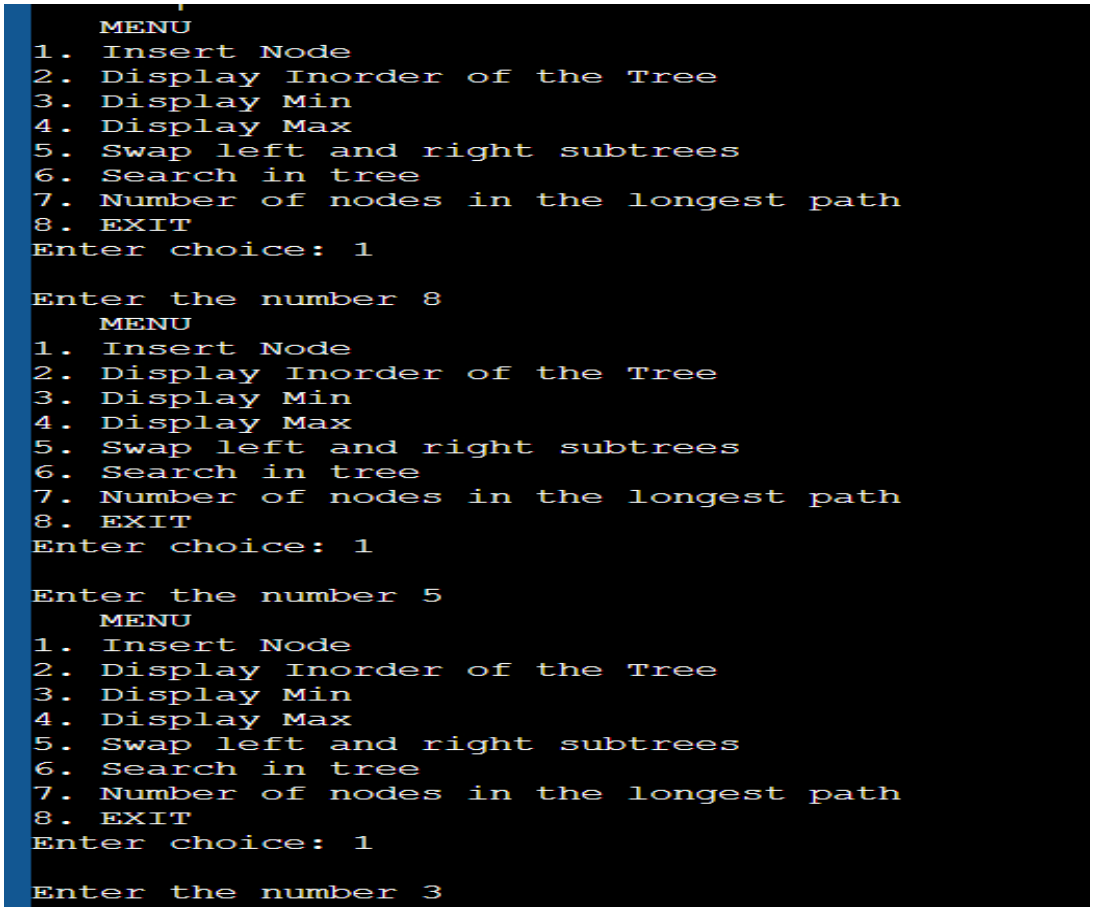
```
switch (choice)
{
case 1:
   cout << "\nEnter the number ";

   cin >> key;
   if (flag == 0)
   {
      root = t1.createTree(key);
      flag = 1;
   }
   else
   {
      t1.insertNode(key, root);
   }
   break;

case 2:
      t1.displayInorder(root);
      break;
case 3:
      t1.displaymin(root);
      break;
case 4:
      t1.displaymax(root);
      break;
case 5:
      t1.swapNodes(root);
      cout<<"Swapped! The new list is : ";
      t1.displayInorder(root);
      break;
case 6:
      cout << "\nEnter the key you want to search: ";
      cin >> searchKey;
      t1.searchNode(searchKey,root);
      break;
```

```
    case 7:
        cout<<"Number of nodes in the longest path is : "<<t1.longestPath(root);
        break;
    case 8:
      exit(0);
    }
  }
  while (choice != 8);
  return 0;
}
```

OUTPUT



```
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 1

Enter the number 6
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 1

Enter the number 12
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 1
```

```
Enter the number 10
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 1

Enter the number 14
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 2
3
5
6
8
10
12
14
```

```
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 3
Minimum number is 3
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 6

Enter the key you want to search: 12
Key found !!!
    MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 7
Number of nodes in the longest path is :3    MENU
```

```
Number of nodes in the longest path is :3   MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 5
Swapped! The new list is : 14
12
10
8
6
5
3
   MENU
1. Insert Node
2. Display Inorder of the Tree
3. Display Min
4. Display Max
5. Swap left and right subtrees
6. Search in tree
7. Number of nodes in the longest path
8. EXIT
Enter choice: 8
```

```
#include <iostream>
#include <string>
#include <cstring>
#include <cstdlib>
using namespace std;
int op;
int cnt=0;
class node
{
  public:
  node *left;
  char word[50],mean[50];
  node *right;
};
class BT
{
  public:
  node *root;
  BT()
  {
  root=NULL;
   }
  void create();
  node* insert(node *,node *);
  void inorder(node *);
  void preorder(node *);
  void postorder(node *);
  void search(node *, char []);
  void modify(node *, char []);
  node *dlt(node * ,char []);
   node *FindMin(node * );
};
void BT::create()
{
  int op;
  node *temp;
```

```
   do
   {
            temp=new node;
            cout<<"Enter A word ";
            cin>>temp->word;
            cout<<"Enter A Meaning : ";
            cin>>temp->mean;
    temp->left=temp->right=NULL;
    if(root==NULL)
    {
             root=temp;
    }
    else
    {
             root=insert(root,temp);
    }
    cout<<"Want to insert again : ";
    cin>>op;
   }while(op==1);
}
node* BT::insert(node *root,node *temp)
{
if(strcmp (temp->word, root->word) < 0 )
 {
  if(root->left == NULL)
   root->left = temp;
  else
   insert(root->left,temp);
 }
 else
 { if(root->right == NULL)
   root->right = temp;
  else
   insert(root->right,temp);
 }
```

```
return root;
}
void BT::inorder(node *temp)
{
   if(temp!=NULL)
   {
            inorder(temp->left);
            cout<<temp->word<<" -> "<<temp->mean<<" , ";
            inorder(temp->right);
   }
}
void BT::preorder(node *temp)
{
   if(temp!=NULL)
   {
            cout<<temp->word<<"-> "<<temp->mean<<" , ";
            preorder(temp->left);
            preorder(temp->right);
   }
}
void BT::postorder(node *temp)
{
   if(temp!=NULL)
   {
              postorder(temp->left);
              postorder(temp->right);
              cout<<temp->word<<"-> "<<temp->mean<<" , ";
   }
}
void BT::search(node *temp , char src[])
{
if(temp != NULL)
{
 if((strcmp(temp->word , src)) == 0)
 {
```

```
    cout<<"\n Word Found ";
    cout<<"\n Word    : "<<temp->word;
    cout<<"\n meaning : "<<temp->mean;
cnt++;
 }
 else
 {
   if((strcmp( src, temp->word )) > 0)
   {
   search(temp->right , src);
  cnt++;
   }
   else
   {
   search(temp->left , src);
  cnt++;
   }
  }
 }
else
  cout<<"\n Word Not Found ";
cout<<"\n Total no of Comparisions to search an element is: "<<cnt;
}
void BT::modify(node *temp , char src[])
{
  if(temp != NULL)
  {
   if((strcmp(temp->word , src)) == 0)
   {
      cout<<"\n Word Found ";
      cout<<"\n Enter New Meaning Of Word "<<temp->word;
      cin>>temp->mean;
   }
   else
   {
   if((strcmp(temp->word , src)) < 0)
```

```
   {
   modify(temp->right , src);
   }
   else if((strcmp(temp->word , src)) > 0)
   {
   modify(temp->left , src);
   }
    }
   }
  else
  cout<<"\n Word Not Found ";
}
node* BT::dlt(node *root , char src[])
{
  if(root != NULL)
  {
     if((strcmp(root->word , src)) > 0)
     {
      root->left = dlt(root->left , src);
     }
     else if((strcmp(root->word , src)) < 0)
  {
  root->right = dlt(root->right , src);
   }
     else
     {
     if(root->left == NULL && root->right == NULL)
     {
       delete(root);
       root = NULL;
     }
     else if(root->left == NULL && root->right!=NULL)
     {
     node *temp = root;
     root = root->right;
     strcpy(root->word , temp->word);
```

```
    strcpy(root->mean , temp->mean);
    temp->right=NULL;
    delete(root);
    }
    else if(root->right == NULL)
    {
    node *temp = root;
    root = root->left;
     strcpy(root->word , temp->word);
    strcpy(root->mean , temp->mean);
    temp->left=NULL;
    delete(root);
    }
    else
    {
    node *temp =  FindMin(root->right);
    strcpy(root->word , temp->word);
    strcpy(root->mean , temp->mean);
     root->right = dlt(root->right , temp->word);
    }
    }
  }
  return root;
}
node* BT:: FindMin(node* root)
{
  while(root->left != NULL) root = root->left;
  return root;
}
int main()
{
  BT b;
  int op;
  char src[100];
  while(1)
  {
```

```
 cout<<"\n ";
 cout<<"\n 1. Insert Binary Search Tree ";
 cout<<"\n 2. Display Inorder,preorder and postorder ";
 cout<<"\n 3. Search The Word ";
 cout<<"\n 4. Modify The Meaning Of Word ";
 cout<<"\n 5. Delete Word From Dictionary ";
 cout<<"\n 6.Exit";
 cout<<"\n Enter your choice:";
 cin>>op;
 switch(op)
 {
 case 1:
b.create();
break;
 case 2:
 cout<<"\n Inorder : ";
b.inorder(b.root);
cout<<"\n Preorder : ";
b.preorder(b.root);
cout<<"\n Postorder : ";
b.postorder(b.root);
break;
 case 3:
 cnt=0;
cout<<"\n Enter The Word Want To Search : ";
 cin>>src;
 b.search(b.root , src);
 break;
 case 4:
 cout<<"\n Enter The Word Want To Modify ";
 cin>>src;
 b.modify(b.root , src);
 break;
 case 5:
 cout<<"\n Enter The Word Want To Delete ";
 cin>>src;
```

```
 b.dlt(b.root , src);
 break;
case 6:
    exit(0);
 break;
 default :
 cout<<"\n Invalid Option ";
 break;
 }
}
}
```

```
 1. Insert Binary Search Tree
 2. Display Inorder,preorder and postorder
 3. Search The Word
 4. Modify The Meaning Of Word
 5. Delete Word From Dictionary
 7.Exit
 Enter your choice:1
Enter A word cpp
Enter A Meaning : high-level
Want to insert again : 1
Enter A word asm
Enter A Meaning : low-level
Want to insert again : 1
Enter A word python
Enter A Meaning : interpreter
Want to insert again : 0


 1. Insert Binary Search Tree
 2. Display Inorder,preorder and postorder
 3. Search The Word
 4. Modify The Meaning Of Word
 5. Delete Word From Dictionary
 7.Exit
 Enter your choice:2

 Inorder : asm -> low-level , cpp -> high-level , python -> interpreter ,
 Preorder : cpp-> high-level , asm-> low-level , python-> interpreter ,
 Postorder : asm-> low-level , python-> interpreter , cpp-> high-level ,
```

```
1. Insert Binary Search Tree
2. Display Inorder,preorder and postorder
3. Search The Word
4. Modify The Meaning Of Word
5. Delete Word From Dictionary
7.Exit
Enter your choice:3

Enter The Word Want To Search : python

Word Found
Word    : python
meaning : interpreter
Total no of Comparisions to search an element is: 1
Total no of Comparisions to search an element is: 2

1. Insert Binary Search Tree
2. Display Inorder,preorder and postorder
3. Search The Word
4. Modify The Meaning Of Word
5. Delete Word From Dictionary
7.Exit
Enter your choice:4

Enter The Word Want To Modify python

Word Found
Enter New Meaning Of Word pythoneasy
```

```
1. Insert Binary Search Tree
2. Display Inorder,preorder and postorder
3. Search The Word
4. Modify The Meaning Of Word
5. Delete Word From Dictionary
7.Exit
Enter your choice:2

Inorder : asm -> low-level , cpp -> high-level , python -> easy ,
Preorder : cpp-> high-level , asm-> low-level , python-> easy ,
Postorder : asm-> low-level , python-> easy , cpp-> high-level ,

1. Insert Binary Search Tree
2. Display Inorder,preorder and postorder
3. Search The Word
4. Modify The Meaning Of Word
5. Delete Word From Dictionary
7.Exit
Enter your choice:5

Enter The Word Want To Delete python


1. Insert Binary Search Tree
2. Display Inorder,preorder and postorder
3. Search The Word
4. Modify The Meaning Of Word
5. Delete Word From Dictionary
7.Exit
Enter your choice:2

Inorder : asm -> low-level , cpp -> high-level ,
Preorder : cpp-> high-level , asm-> low-level ,
Postorder : asm-> low-level , cpp-> high-level ,
```

```
#include<iostream>
#include<cstring>
//int mod = 1e9+7;
using namespace std;
class Telephone {
  unsigned long long key;
  int address;
  int num;
  unsigned long long mobile[10];
  string name[10];

public:
  Telephone() {
  for (int i = 0; i < 10; i++) {
  mobile[i] = 0;
  }
  for (int i = 0; i < 10; i++) {
  name[i] = "-";
  }
  }
  void insert_record() {
  cout << "Enter no of records you want to enter :- ";
  cin >> num;
  cout << endl;
  cout << "Which Collision handling technique do you want to use"<< endl << "1. Linear Probing" << endl << "2. Quadratic Probing" << endl;
  int flag;
  cout << "Enter your Choice :- ";
  cin >> flag;
  cout << endl;
  while (num--) {
  cout << "Enter Telephone no :- ";
  cin >> key;
  cout << endl;
  address = hash_function(key);
```

```
if (mobile[address] == 0) {
mobile[address] = key;
cout << "Enter name of the person :- ";
cin >> name[address];
cout << endl;
}
else if (flag == 1) {
hash_collision_linear_probing(mobile, name, key);
}
else if (flag == 2) {
hash_collision_quadratic_probing(mobile, name, key);
}
}
}

void hash_collision_linear_probing(unsigned long long mobile[], string name[], unsigned long long key) {
int adr = hash_function(key);
while (mobile[(adr % 10)] != 0) {
adr++;
}
mobile[adr % 10] = key;
cout << "Enter name of the person :- ";
cin >> name[adr % 10];
cout << endl;
void hash_collision_quadratic_probing(unsigned long long mobile[], string name[], unsigned long long key) {
int adr = hash_function(key);
int i = 1;
while (mobile[(adr % 10)] != 0) {
adr += (i * i);
i++;
}
mobile[adr % 10] = key;
cout << "Enter name of the person :- ";
cin >> name[adr % 10];
cout << endl;
}
```

```
  void display() {
  cout << "Index\tName\tMobile" << endl;
  for (int i = 0; i < 10; i++) {
  cout << i << "\t" << name[i] << "\t" << mobile[i] << endl;
  }
  }
  int hash_function(unsigned long long key) {
  return key % 10;
  }
};

int main() {
  Telephone t1;
  int choice;
  char ch;

  do {
  cout << "*Telephone Directory*" << endl;
  cout << "1. Insert record in Directory" << endl;
  cout << "2. Display Telephone Directory" << endl;
  cout << "3. Exit" << endl;
  a
  cout << endl << "Enter your choice :- ";
  cin >> choice;
  cout << endl;
  switch (choice) {
  case 1:
  t1.insert_record();
  break;
  case 2:
  t1.display();
  break;
  }
  } while (choice < 3);
  cout << "Thanks for Using My software" << endl;
  return 0;
```

}

```
ubuntu@HW-LAB:~/Desktop/c++$ ./ass
****Telephone Directory****
1. Insert record in Directory
2. Display Telephone Directory
3. Exit

Enter your choice :- 1

Enter no of records you want to enter :- 3

Which Collision handling technique do you want to use
1. Linear Probing
2. Quadratic Probing
Enter your Choice :- 1

Enter Telephone no :- 744

Enter name of the person :- Sid

Enter Telephone no :- 744

Enter name of the person :- Harsh

Enter Telephone no :- 744

Enter name of the person :- Nish
```

```
Enter your choice :- 2

Index   Name    Mobile
0       -       0
1       -       0
2       -       0
3       -       0
4       Sid     744
5       Harsh   744
6       Nish    744
7       -       0
8       -       0
9       -       0
****Telephone Directory****
1. Insert record in Directory
2. Display Telephone Directory
3. Exit

Enter your choice :-
```

```
Enter your Choice :- 2

Enter Telephone no :- 744

Enter name of the person :- sid

Enter Telephone no :- 744

Enter name of the person :- harh

Enter Telephone no :- 744

Enter name of the person :- nish

****Telephone Directory****
1. Insert record in Directory
2. Display Telephone Directory
3. Exit

Enter your choice :- 2

Index   Name    Mobile
0       -       0
1       -       0
2       -       0
3       -       0
4       sid     744
5       harh    744
6       -       0
7       -       0
8       -       0
9       nish    744
****Telephone Directory****
1. Insert record in Directory
2. Display Telephone Directory
3. Exit

Enter your choice :-
```