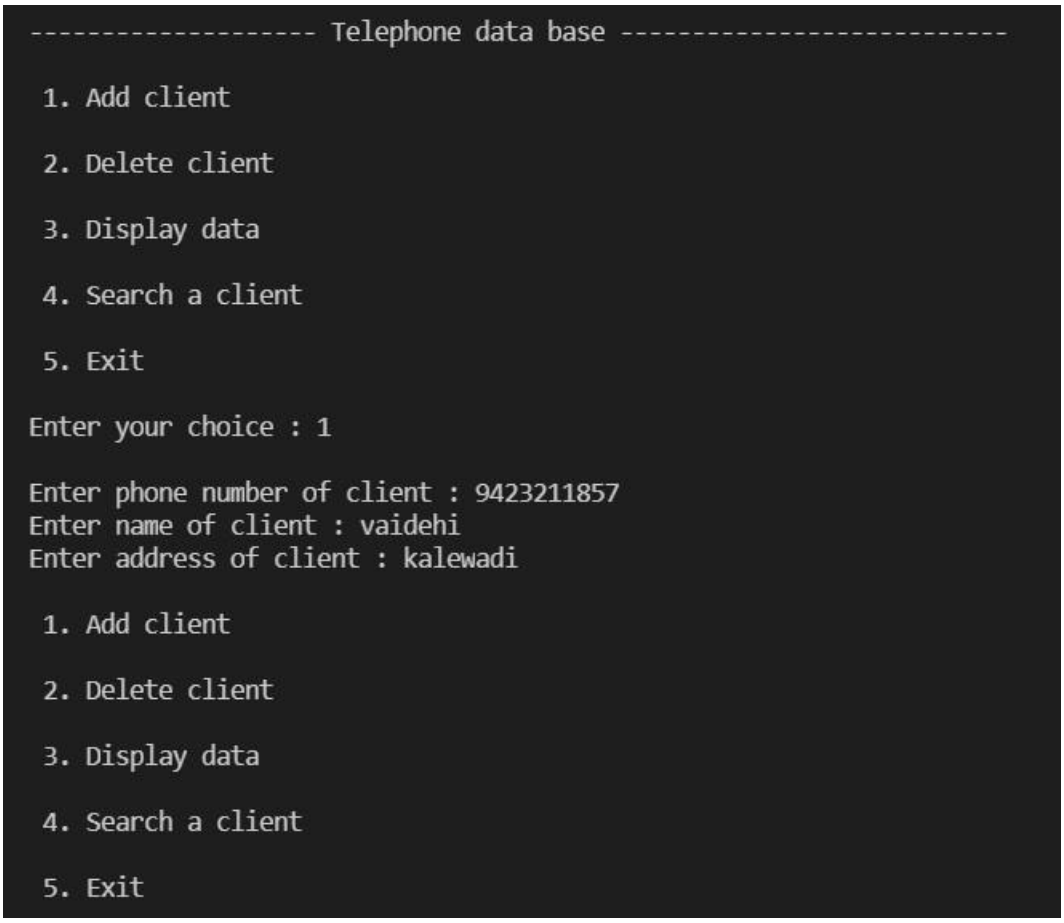
- print("\n-------------------- Telephone data base ---------------------------")

```
telephonediary = [] namediary = []
addressdiary = []
for i in range(10):
   a = []
   telephonediary.append(a)     namediary.append(a)
addressdiary.append(a)

def add_client(namediary, addressdiary, telephonediary, count):
   phn = int(input("\nEnter phone number of client : "))     naam =
input("Enter name of client : ")     addss = input("Enter address of
client : ")
   hashkey = phn % 10

   for i in range(len(telephonediary)):
  if i == hashkey:
         if telephonediary[i] != []:
            b = []
            for j in telephonediary[i]:
               b.append(j)
            b.append(phn)              telephonediary.insert(i,
b)              telephonediary.pop(i+1)
      else:
      b = []

            b.append(phn)              telephonediary.insert(i,
b)              telephonediary.pop(i+1)
            break
   for i in range(len(namediary)):
  if i == hashkey:
 if namediary[i] != []:
            b = []
  for j in namediary[i]:
               b.append(j)
            b.append(naam)
  namediary.insert(i, b)
namediary.pop(i+1)
      else:
b = []
        b.append(naam)
 namediary.insert(i, b)
 namediary.pop(i+1)
        break
  for i in range(len(addressdiary)):
    if i == hashkey:
if addressdiary[i] != []:
        b = []
  for j in addressdiary[i]:
          b.append(j)
        b.append(addss)
addressdiary.insert(i, b)
addressdiary.pop(i+1)
else:
b = []
        b.append(addss)
addressdiary.insert(i, b)
addressdiary.pop(i+1)
 break
def delete_client(telephonediary, namediary, addressdiary, dele, count):
  global  pn      pn  =  count      for  i  in
range(len(telephonediary)):
    if (telephonediary[i] != []):           for j in
range(len(telephonediary[i])):         if (dele ==
telephonediary[i][j]):
          pn = pn - 1


          for k in range(len(namediary)):
for l in range(len(namediary[k])):
              if (j == l and i == k):
                print("\n---",namediary[k][l]," has been deleted.---")
namediary[k].pop(l)                  break
```

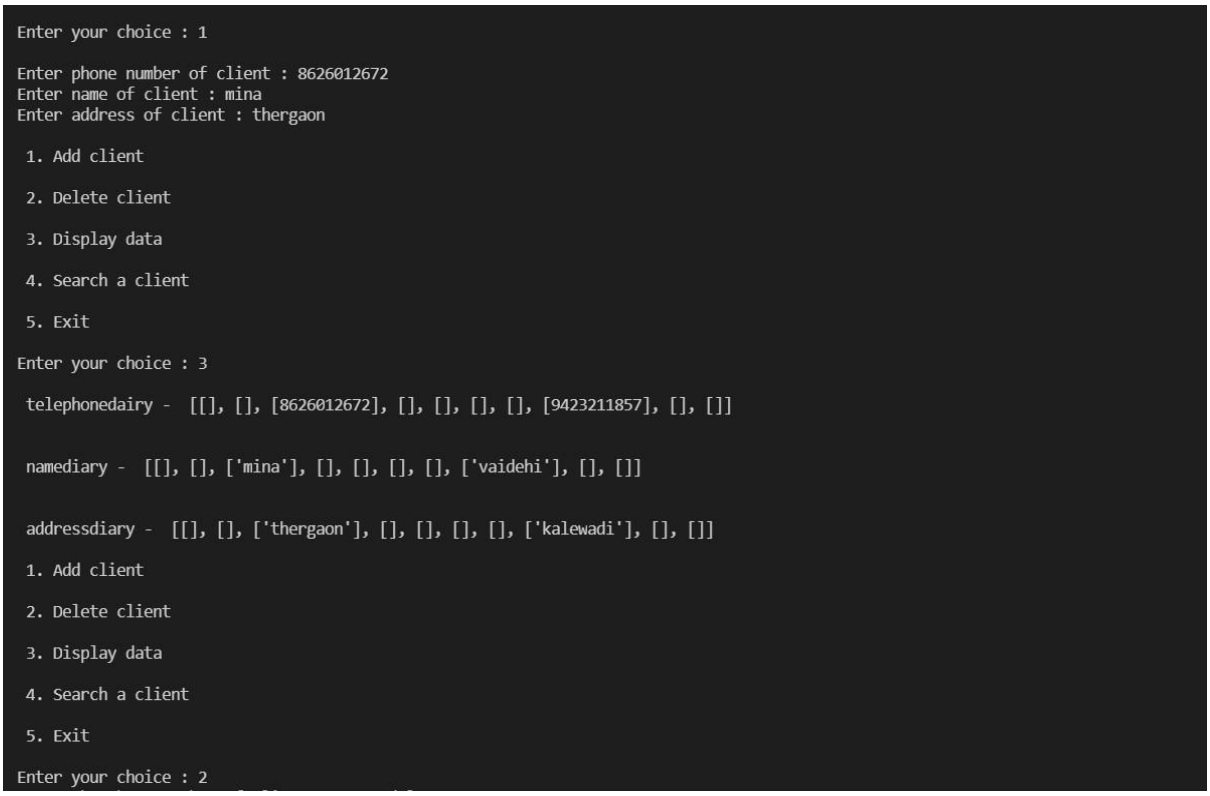
```
          for k in range(len(addressdiary)):
for l in range(len(addressdiary[k])):               if (j == l
and i == k):                    addressdiary[k].pop(l)
break

          telephonediary[i].pop(j)
break     else:      pass

  if (pn==count):
    print("\n-----Number to be deleted not in the client data-----")




def search_number(namediary, telephonediary, addressdiary, sea, count):
  global  dn      dn  =  count      for  i  in
range(len(telephonediary)):
    if (telephonediary[i] != []):      for j in
range(len(telephonediary[i])):          if (sea ==
telephonediary[i][j]):
          dn = dn - 1

          print("\n---------Client found---------")          print("Number -
",telephonediary[i][j])

          for k in range(len(namediary)):
for l in range(len(namediary[k])):
              if (j == l and i == k):
                print("Name - ",namediary[k][l])
                break

          for k in range(len(addressdiary)):
for l in range(len(addressdiary[k])):               if (j == l
and i == k):
                print("Address - ",addressdiary[k][l])
```

```
                                break

    if (dn == count):
        print("\n-----Number to be searched not found-----")



def display(namediary, addressdiary, telephonediary):
    print("\n telephonedairy - ",telephonediary)     print("\n\n
namediary - ",namediary)     print("\n\n addressdiary -
",addressdiary)



def main():     count
= 0     while True:
        print("\n 1. Add client ")         print("\n 2.
Delete client")         print("\n 3. Display data")
print("\n 4. Search a client")         print("\n 5.
Exit")

        ch = int(input("\nEnter your choice : "))

        if (ch == 5):              print ("End of
Program")              break

        elif (ch == 1):
            count = count + 1            add_client(namediary, addressdiary,
telephonediary, count)

        elif (ch == 2):
            dele = int(input("Enter the phone number of client u wanna delete : "))            delete_client(telephonediary,
namediary, addressdiary, dele, count)
            count = pn



        elif(ch == 3):
            display(namediary, addressdiary, telephonediary)
    elif(ch == 4):
        sea = int(input("Enter the telephone number of client u wanna seach : "))            search_number(namediary,
telephonediary, addressdiary, sea, count)

else:
        print("Wrong choice entered")

main()
```

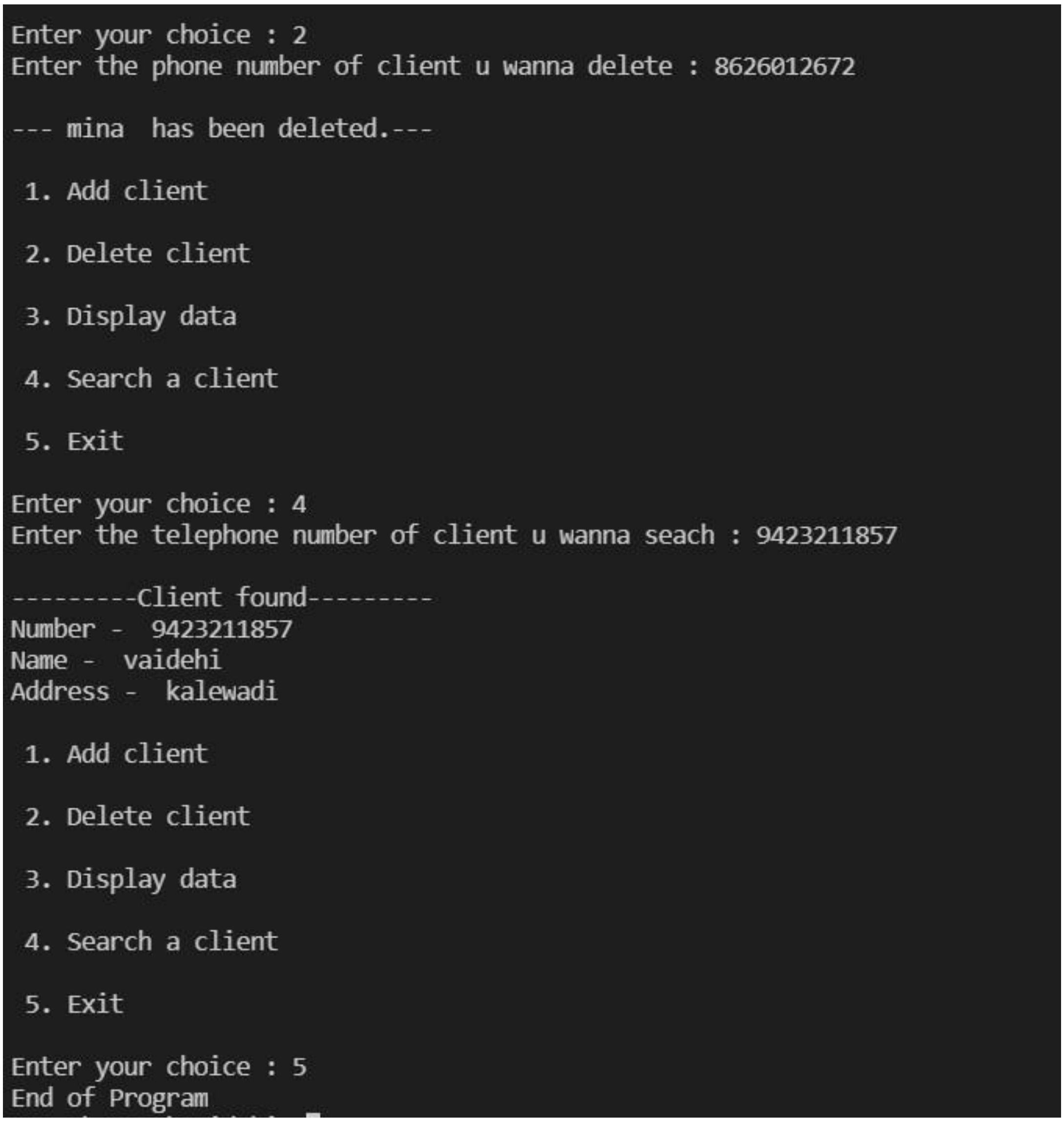
OUTPUT:



```
 -------------------- Telephone data base ---------------------------

  1. Add client

  2. Delete client

  3. Display data

  4. Search a client

  5. Exit

 Enter your choice : 1

 Enter phone number of client : 9423211857
 Enter name of client : vaidehi
 Enter address of client : kalewadi

  1. Add client

  2. Delete client

  3. Display data

  4. Search a client

  5. Exit
```



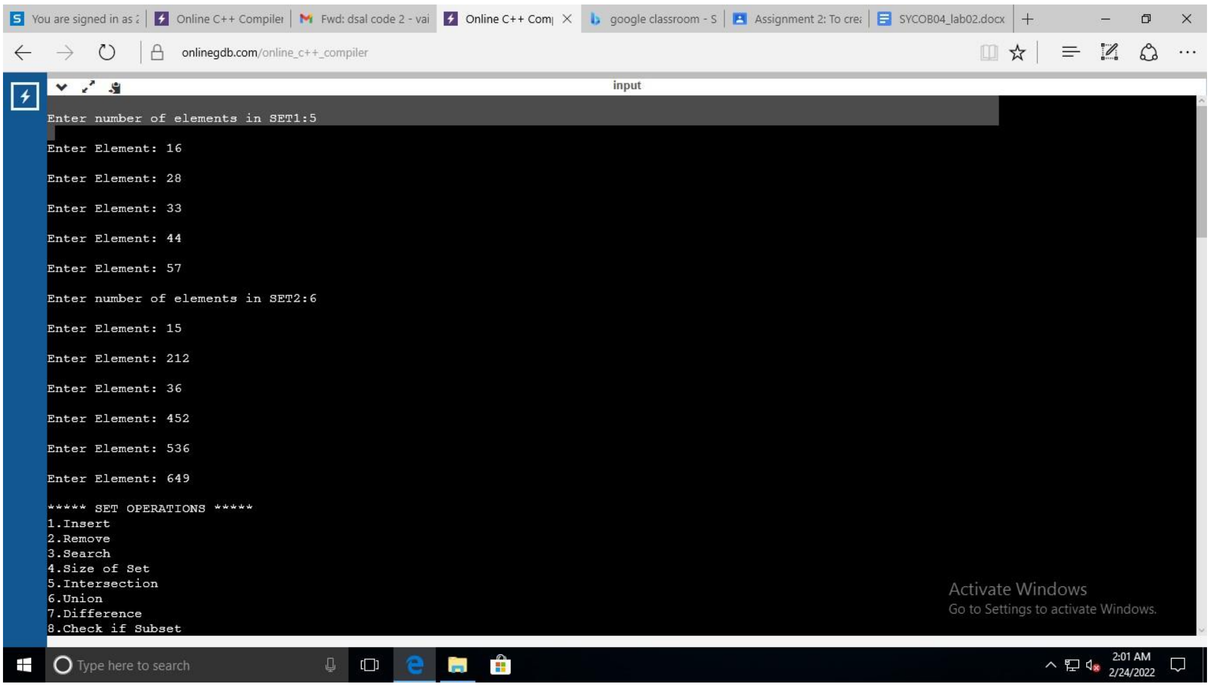
```
Enter your choice : 1

Enter phone number of client : 8626012672
Enter name of client : mina
Enter address of client : thergaon

 1. Add client

 2. Delete client

 3. Display data

 4. Search a client

 5. Exit

Enter your choice : 3

 telephonedairy -  [[], [], [8626012672], [], [], [], [], [9423211857], [], []]


 namediary -  [[], [], ['mina'], [], [], [], [], ['vaidehi'], [], []]


 addressdiary -  [[], [], ['thergaon'], [], [], [], [], ['kalewadi'], [], []]

 1. Add client

 2. Delete client

 3. Display data

 4. Search a client

 5. Exit

Enter your choice : 2
```



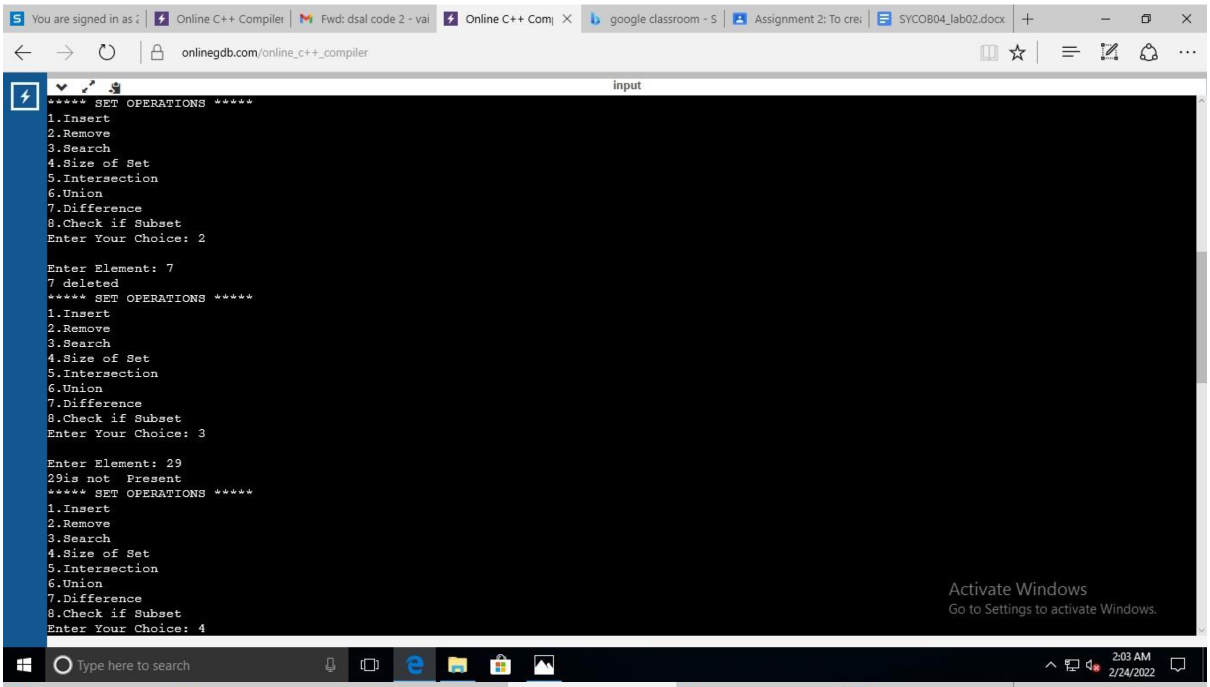
```
Enter your choice : 2
Enter the phone number of client u wanna delete : 8626012672

--- mina  has been deleted.---

 1. Add client

 2. Delete client

 3. Display data

 4. Search a client

 5. Exit

Enter your choice : 4
Enter the telephone number of client u wanna seach : 9423211857

---------Client found---------
Number -  9423211857
Name -  vaidehi
Address -  kalewadi

 1. Add client

 2. Delete client

 3. Display data

 4. Search a client

 5. Exit

Enter your choice : 5
End of Program
```

```
/*Assignment no: 02
Assignment 2: To create ADT that implement the "set" concept. a. Add
(newElement) -Place a value into the set
b. Remove (element) Remove the value
c. Contains (element) Return true if element is in collection
d. Size () Return number of values in collection Iterator () Return an iterator used to loop
   over collection
e. Intersection of two sets
f. Union of two sets
g. Difference */

#include  <iostream>  using
namespace  std;  const  int
MAX=50;    template<class
T>
class SET
{
 T data[MAX];
int n; public:
SET()
{
  n=-1;
}
 bool insert(T);  bool
remove(T);  bool
contains(T);  int
size();  void print();
 void input(int num);
 SET unionS(SET,SET);
 SET intersection(SET,SET);  SET
difference(SET,SET);
 bool subset(SET);
};
template<class T>
bool SET<T>::subset(SET<T> s2)
{
 int count=0;  int
size=s2.size();
 for(int i=0;i<=n;i++)
 {
  for(int j=0;j<=s2.n;j++)
  {
   if(data[i]==s2.data[j])
   {
             count++;
             break;
   }
  }
 }
 if(count==size)
 {
  return true;
 }
 return false;
}
template<class T>
void SET<T>::input(int num)
{
 T element;  for(int
i=0;i<num;i++)
 {
  cout<<"\nEnter Element: "<<i+1;   cin>>element;
  insert(element);
 }
}
template<class T>
void SET<T>::print()
{
 for(int i=0;i<=n;i++)
  cout<<" "<<data[i];
}
template<class T>
SET<T> SET<T>::unionS(SET<T> s1,SET<T> s2)
{
 SET<T> s3;
 int   flag=0;      int   i=0;
for(i=0;i<=s1.n;i++)
 {
  s3.insert(s1.data[i]);
 }
 for(int j=0;j<=s2.n;j++)
 {
  flag=0;   for(i=0;i<=s1.n;i++)
  {
   if(s1.data[i]==s2.data[j])
   {
            flag=1;   break;
   }
  }
  if(flag==0)
  {
   s3.insert(s2.data[j]);
  }
 }
 return s3;
}
template<class T>
SET<T> SET<T>::difference(SET<T> s1,SET<T> s2)
{
 SET<T> s3;  int
flag=1;
 for(int i=0;i<=s1.n;i++)
 {
  for(int j=0;j<=s2.n;j++)
  {
   if(s1.data[i]==s2.data[j])
   {
            flag=0;   break;
   }
   else flag=1;
  }
  if(flag==1)
  {
   s3.insert(s1.data[i]);
  }
 }
 return s3;
}
template<class T>
SET<T> SET<T>::intersection(SET<T> s1,SET<T> s2)
{
 SET<T> s3;
 for(int i=0;i<=s1.n;i++)
 {
  for(int j=0;j<=s2.n;j++)
  {
   if(s1.data[i]==s2.data[j])
   {
                 s3.insert(s1.data[i]);
               break;
   }
  }
 }
 return s3;
}
template<class T> bool
SET<T>::insert(T element)
{
 if(n>=MAX-1)
 {
  cout<<"\nOverflow.SET is full.\n";
  return false;
 }
 data[++n]=element;
 return true;
}
template<class T>
bool SET<T>::remove(T element)
{
 if(n==-1)
 {
  cout<<"Underflow. Cannot perform delete operation on empty SET.";   return false;
 }
 for(int i=0;i<=n;i++)
 {
  if(data[i]==element)
  {
    for(int j=i;j<n;j++)
   {
                data[j]=data[j+1];
   }

   return true;
  }
 }
 //data[n--]=0;
 return false;
}
template<class T>
bool SET<T>::contains(T element)
{
 for(int i=0;i<=n;i++)
 {
  if(data[i]==element)
   return true;
 }
 return false;
}
template<class T>
int SET<T>::size()
{
 return n+1;
}
int main() {
 SET<int> s1,s2,s3;  int choice;  int element;
cout<<"\nEnter number of elements in SET1:";
cin>>element;//element is used for taking size
s1.input(element);  cout<<"\nEnter number of elements in
SET2:";  cin>>element;//element is used for taking size
s2.input(element);
 do
 {
  cout<<"\n***** SET OPERATIONS *****"
                 <<"\n1.Insert"
                  <<"\n2.Remove"
                 <<"\n3.Search"
                  <<"\n4.Size of Set"
                  <<"\n5.Intersection"
                 <<"\n6.Union"
                  <<"\n7.Difference"
 <<"\n8.Check if Subset"  <<"\nEnter Your Choice: ";
  cin>>choice;
  switch(choice)
  {
  case 1:
   cout<<"\nEnter Element: ";    cin>>element;
if(s1.insert(element))
    {
                   cout<<element<<" inserted";
   }
   else
   {
                  cout<<"Insertion Failed";
   }
   break;   case 2:    cout<<"\nEnter
Element: ";    cin>>element;
if(s1.remove(element))
   {
                   cout<<element<<" deleted";
   }
   else
   {
                  cout<<"Deletion Failed";
   }
   break;   case 3:    cout<<"\nEnter
Element: ";    cin>>element;
   if(s1.contains(element))
   {
                   cout<<element<<" is present";
   }
   else
   {
                    cout<<element<<"is not  Present";
   }
   break;   case 4:    cout<<"\nSize =
"<<s1.size();    break;   case 5:
s3=s1.intersection(s1,s2);
   cout<<"\nSET 1's elements: ";
s1.print();    cout<<"\nSET 2's
elements: ";    s2.print();
cout<<"\nIntersection: :";    s3.print();
break;   case 6:
   s3=s1.unionS(s1,s2);
   cout<<"\nSET 1's elements: ";
s1.print();    cout<<"\nSET 2's
elements: ";    s2.print();
cout<<"\nUnion :";    s3.print();    break;
case 7:    s3=s1.difference(s1,s2);
cout<<"\nSET 1's elements: ";
   s1.print();
   cout<<"\nSET 2's elements: ";    s2.print();
   cout<<"\nDifference :";
   s3.print();    break;
case 8:
if(s1.subset(s2))
  {
   cout<<"\nS2 is Subset of S1.";
  }
  else
  {
   cout<<"\nNot Subset.";
  }
  break;
  }
 }while(choice!=0);  return 0;
}
```

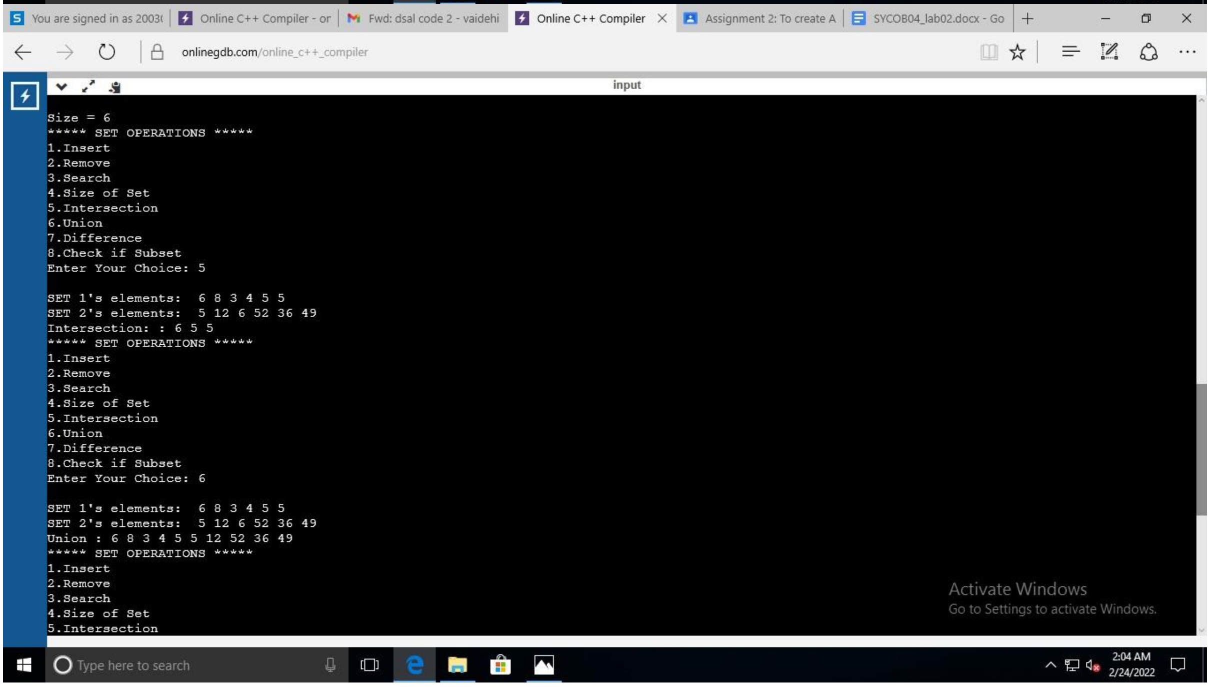
OUTPUT:



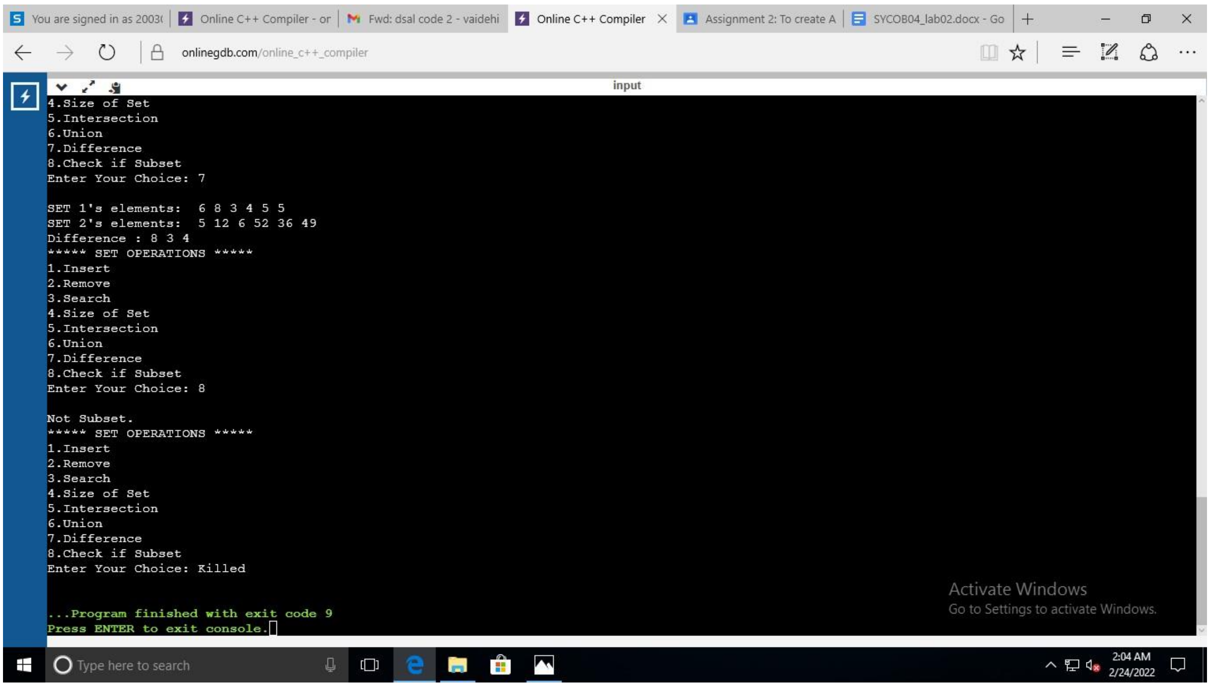
```
Enter number of elements in SET1:5

Enter Element: 16

Enter Element: 28

Enter Element: 33

Enter Element: 44

Enter Element: 57

Enter number of elements in SET2:6

Enter Element: 15

Enter Element: 212

Enter Element: 36

Enter Element: 452

Enter Element: 536

Enter Element: 649

***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
```



```
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 2

Enter Element: 7
7 deleted
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 3

Enter Element: 29
29is not  Present
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 4
```


You are signed in as 2003
Online C++ Compiler - or
Fwd: dsal code 2 - vaidehi
Online C++ Compiler
Assignment 2: To create A
SYCOB04_lab02.docx - Go
onlinegdb.com/online_c++_compiler
input
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:04 AM
2/24/2022

```
Size = 6
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 5

SET 1's elements:  6 8 3 4 5 5
SET 2's elements:  5 12 6 52 36 49
Intersection: : 6 5 5
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 6

SET 1's elements:  6 8 3 4 5 5
SET 2's elements:  5 12 6 52 36 49
Union : 6 8 3 4 5 5 12 52 36 49
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
```



You are signed in as 2003(
Online C++ Compiler - or
Fwd: dsal code 2 - vaidehi
Online C++ Compiler
Assignment 2: To create A
SYCOB04_lab02.docx - Go
onlinegdb.com/online_c++_compiler
input
Activate Windows
Go to Settings to activate Windows.
Type here to search
2:04 AM
2/24/2022

```
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 7

SET 1's elements:  6 8 3 4 5 5
SET 2's elements:  5 12 6 52 36 49
Difference : 8 3 4
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: 8

Not Subset.
***** SET OPERATIONS *****
1.Insert
2.Remove
3.Search
4.Size of Set
5.Intersection
6.Union
7.Difference
8.Check if Subset
Enter Your Choice: Killed


...Program finished with exit code 9
Press ENTER to exit console.
```

Assignment No. 3

Problem Statement :

A book consists of chapters, chapters consist of sections and sections consist of subsections.Construct a tree and print the nodes. Find the time and space requirements of your method.

Program

```
#include<bits/stdc++.h>                 using
namespace std;

struct node
{
   char  label[60];         int
chcount;                    node
*child[50];
}*root;

class general
{     public:
      void              insert();
void                    display();
general()
      {
         root == NULL;
      }
};

void general::insert()
{          int  secount;          root  =  new  node();
cout<<"Enter the name of book : ";     cin>>root-
>label;
   cout<<"Enter  the  total  number  of  chapters  in  book  :  ";       cin>>root-
>chcount;     for(int i=0;i<root->chcount;i++)
   {
      root->child[i] = new node();        cout<<"Enter the name
of chapters : ";        cin>>root->child[i]->label;
```

```
        cout<<"Enter the number of sections : ";          cin>>root-
>child[i]->chcount;                        for(int  j=0;j<root->child[i]-
>chcount;j++)
        {
            root->child[i]->child[j] = new node();
cout<<"Enter the name of section : ";             cin>>root-
>child[i]->child[j]->label;

            cout<<"Enter the number of sub sections : ";             cin>>root->child[i]-
>child[j]->chcount;              for(int k=0; k<root->child[i]->child[j]->chcount; k++)
            {
                root->child[i]->child[j]->child[k]          =          new          node();
cout<<"Enter the name of sub section : ";                      cin>>root->child[i]-
>child[j]->child[k]->label;             }

        }
    }
}

void general::display()
{
    if(root != NULL)
    {
        cout<<"********** Hierarchy of Book **********"<<endl;         cout<<"Book
Name is : "<<root->label<<endl;         for(int i=0; i<root->chcount; i++)
        {
            cout<<"-- "<<root->child[i]->label<<endl;            for(int j=0; j<root-
>child[i]->chcount; j++)
            {
                cout<<"---- "<<root->child[i]->child[j]->label<<endl;                  for(int k=0; k<root-
>child[i]->child[j]->chcount; k++)
                {
                    cout<<"------ "<<root->child[i]->child[j]->child[k]->label<<endl;
                }
            }
        }
    }
}
```

```
int        main()        {
general  tree;       int
ch;

   do
   {
      cout<<"Press 1 to Insert."<<endl
         <<"Press 2 to Display."<<endl
         <<"Press 3 to exit."<<endl            <<"Enter the
choice:"<<endl;       cin>>ch;


      switch(ch)
      {           case 1:
tree.insert();                break;
case 2:
tree.display();               break;
case 3:               return 0;
}
   }while(ch<4);     return 0;
}
```

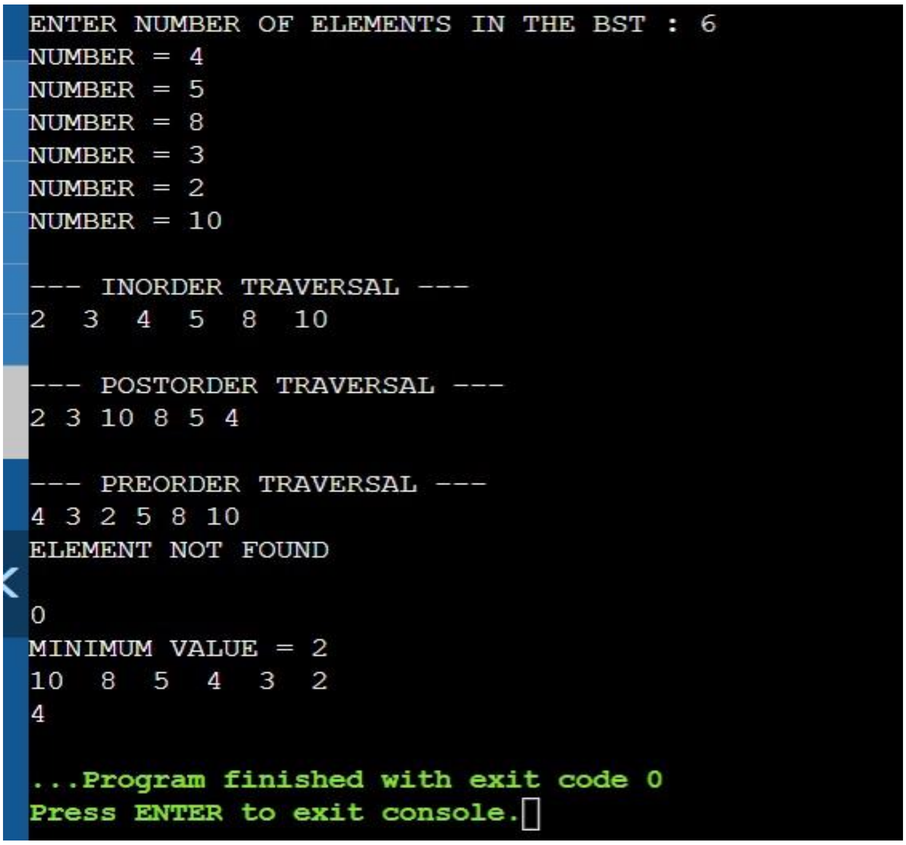
```
/*Assignment No 4 Beginning with an empty binary search tree, construct binary search tree by inserting the values in the order given. After constructing a binary tree - i. Insert new node ii. Find number of nodes in longest path from root iii. Minimum data value found in the tree iv. Change a tree so that the roles of the left and right pointers are swapped at every node v. Search a value*/


#include<iostream>

#include<math.h> using

namespace std;


struct Bstnode

{

 int data;

 Bstnode *left = NULL;

 Bstnode *right = NULL;


};


class Btree

{


  int n;   int

x;   int flag;


public:

  Bstnode * root;

 Btree()

 {

  root = NULL;

}


Bstnode *GetNewNode(int in_data)

 {

  Bstnode * ptr = new Bstnode();    ptr-

>data = in_data;   ptr->left = NULL;   ptr-

>right = NULL;   return ptr;

 }
```

```
Bstnode *insert( Bstnode *temp , int in_data)
{
 if( temp == NULL )
 {
  temp = GetNewNode(in_data);
 }
 else if( temp->data > in_data)
 {
  temp->left = insert(temp->left , in_data);
 }
 else
 {
  temp->right = insert( temp->right , in_data);
 }
 return temp;
}

void input()
{
 cout<<"ENTER NUMBER OF ELEMENTS IN THE BST : ";
 cin>>n;   for(int i = 0 ; i < n ;
i++)
 {
  cout<<"NUMBER = ";
  cin>>x;    root = insert(root
, x);
 }
}

int search(Bstnode *temp ,int in_data)
{
 if( temp != NULL)
 {
  if(temp->data == in_data)
  {
   cout<<":-- RECORD FOUND --:"<<endl;
```

```
    return 1;
   }
   else if(in_data < temp->data)
   {
    this->search(temp->left, in_data);
   }
   else if(in_data > temp->data)
   {
    this->search(temp->left , in_data);
   }
  }
  else
  {
    return 0;
  }
}

 void minvalue(Bstnode *temp)
 {
  while(temp->left != NULL)
  {
   temp = temp->left;
  }
  cout<<"MINIMUM VALUE = "<<temp->data<<endl;
 }
 void mirror(Bstnode *temp)
 {
  if(temp == NULL)
  {
   return;
  }
  else
  {
   Bstnode *ptr;    mirror(temp-
>left);    mirror(temp->right);
```

```
ptr = temp->left;    temp->left =
temp->right;    temp->right = ptr;
  }
 }

 void display()
 {
  cout<<endl<<"--- INORDER TRAVERSAL ---"<<endl;
inorder(root);   cout<<endl;
  cout<<endl<<"--- POSTORDER TRAVERSAL ---"<<endl;
postorder(root);   cout<<endl;   cout<<endl<<"--- PREORDER
TRAVERSAL ---"<<endl;   preorder(root);   cout<<endl;

 }

 void inorder(Bstnode *temp)
 {
  if(temp != NULL)
  {
   inorder(temp->left);    cout<<temp->data<<"  ";
inorder(temp->right);
  }
 }

 void postorder(Bstnode *temp)
 {
  if(temp != NULL)
  {
   postorder(temp->left);    postorder(temp->right);
cout<<temp->data<<" ";
  }
 }

void preorder(Bstnode *temp)
{
  if(temp != NULL)
 {
  cout<<temp->data<<" ";    preorder(temp->left);
preorder(temp->right);
 }
 }

 int depth(Bstnode *temp)
 {
  if(temp   ==   NULL)               return   0;         return   (max((depth(temp-
>left)),(depth(temp->right))) +1);
 }
};
int main()
{
 Btree obj;  obj.input();
obj.display();  int a = 0;  a =
obj.search(obj.root,10);
 if( a == 0)
 {
  cout<<"ELEMENT NOT FOUND"<<endl;
 }
 else
  cout<<"ELEMENT FOUND"<<endl;
cout<<endl<<a<<endl;  obj.minvalue(obj.root);
obj.mirror(obj.root);  obj.inorder(obj.root);
 //int d ;
 cout<<endl<<obj.depth(obj.root);
//cout<<endl<<d<<endl;  return 0;
}
```

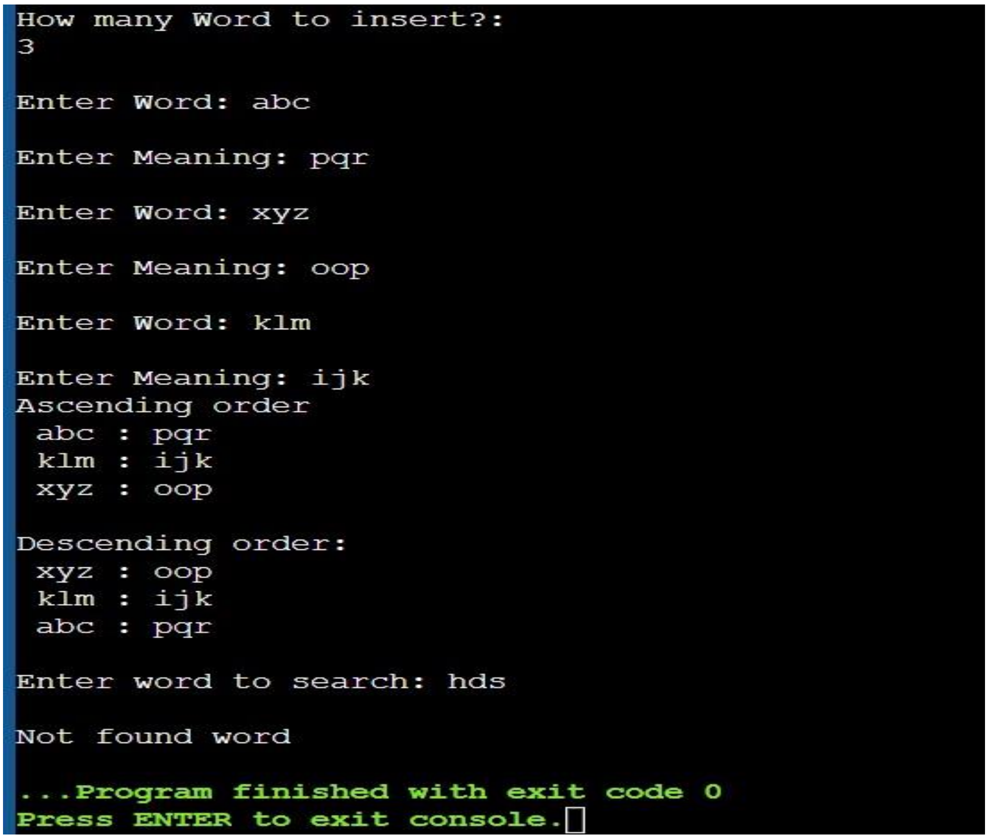
OUTPUT:



```
ENTER NUMBER OF ELEMENTS IN THE BST : 6
NUMBER = 4
NUMBER = 5
NUMBER = 8
NUMBER = 3
NUMBER = 2
NUMBER = 10

--- INORDER TRAVERSAL ---
2  3  4  5  8  10

--- POSTORDER TRAVERSAL ---
2 3 10 8 5 4

--- PREORDER TRAVERSAL ---
4 3 2 5 8 10
ELEMENT NOT FOUND

0
MINIMUM VALUE = 2
10  8  5  4  3  2
4

...Program finished with exit code 0
Press ENTER to exit console.
```

```
/*ASSIGNMENT NO 5:

A Dictionary stores keywords & its meaning. Provide facility for adding new keywords, deleting keywords, updating
values of any entry. Provide facility to display whole data sorted in ascending/ Descending order. Also find how
many maximum comparisons may require for finding any keyword. Use Binary Search Tree for implementation.*/


#include <iostream>
#include<string> using
namespace std; class
dictionary;
class node
{
 string word,meaning;   node
*left,*right;   public:      friend
class dictionary;
 node()
 {
  left=NULL;
  right=NULL;

 }
 node(string word, string meaning)
 {
  this->word=word;   this->meaning=meaning;
left=NULL;
  right=NULL;
 }
};

class dictionary
{
 node *root; public:
 dictionary()
{
  root=NULL;
}
 void create();
 void inorder_rec(node *rnode);  void
postorder_rec(node *rnode);  void inorder()
 {
  inorder_rec(root);
 }
 void postorder();

 bool insert(string word,string meaning);  int
search(string key);
 };
int dictionary::search(string key)
{
 node *tmp=root;  int
count;
 if(tmp==NULL)
 {
  return -1;
 }
 if(root->word==key)
  return 1;
 while(tmp!=NULL)
 {

  if((tmp->word)>key)
  {
   tmp=tmp->left;
   count++;
  }
  else if((tmp->word)<key)
  {
   tmp=tmp->right;
   count++;
  }
  else if(tmp->word==key)
  {
   return ++count;
  }
 }
 return -1;

}
void dictionary::postorder()
{
 postorder_rec(root);
}
void dictionary::postorder_rec(node *rnode)
{
 if(rnode)
 {
  postorder_rec(rnode->right);
  cout<<" "<<rnode->word<<" : "<<rnode->meaning<<endl;   postorder_rec(rnode->left);
 }
}
void dictionary::create()
{
 int n;
 string wordI,meaningI;  cout<<"\nHow many
Word to insert?:\n";
 cin>>n;  for(int
i=0;i<n;i++)
 {
  cout<<"\nENter Word: ";
cin>>wordI;   cout<<"\nEnter
Meaning: ";   cin>>meaningI;
  insert(wordI,meaningI);
 }
}
void dictionary::inorder_rec(node *rnode)
{
 if(rnode)
 {
  inorder_rec(rnode->left);
  cout<<" "<<rnode->word<<" : "<<rnode->meaning<<endl;   inorder_rec(rnode->right);
 }
}
bool dictionary::insert(string word, string meaning)
{
 node *p=new node(word, meaning);
 if(root==NULL)
 {
  root=p;
  return true;
 }
 node *cur=root;  node
*par=root;
 while(cur!=NULL) //traversal
 {
  if(word>cur->word)   {par=cur;
cur=cur->right;
  }
  else if(word<cur->word)
  {
   par=cur;    cur=cur->left;
  }
  else
  {
   cout<<"\nWord is already in the dictionary.";    return false;
  }
 }
 if(word>par->word) //insertion of node
 {
  par->right=p;
  return true;
 }
 else
 {
  par->left=p;

  return true;
 } }

int main() {
 string word;  dictionary months;
months.create();
cout<<"Ascending order\n";
 months.inorder();

 cout<<"\nDescending order:\n";
 months.postorder();

 cout<<"\nEnter word to search: ";  cin>>word;
 int comparisons=months.search(word);  if(comparisons==-1)
 {
  cout<<"\nNot found word";
 }
 else
 {
  cout<<"\n "<<word<<" found in "<<comparisons<<" comparisons";
 }
 return 0;
}
```

OUTPUT:



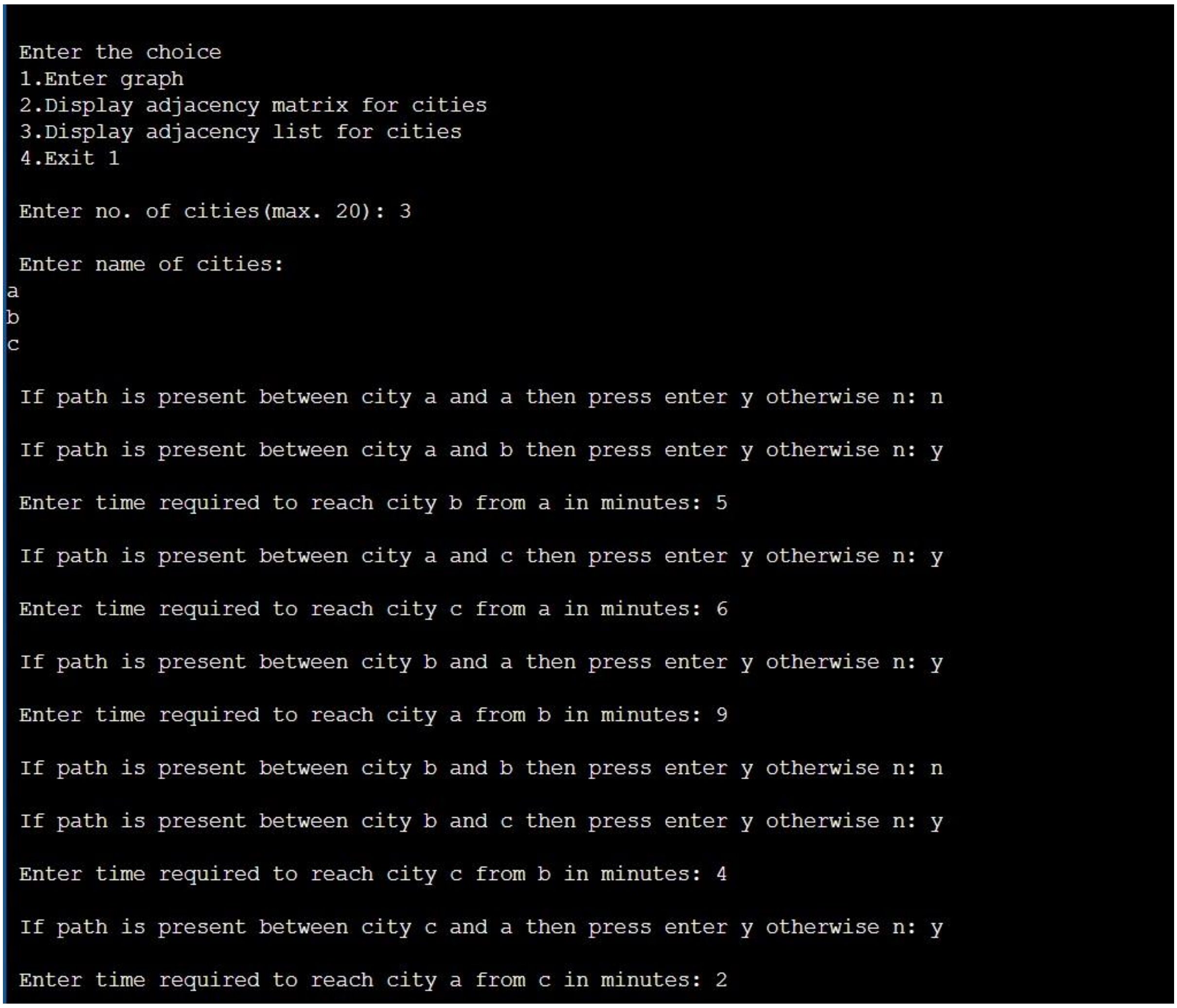
```
How many Word to insert?:
3

Enter Word: abc

Enter Meaning: pqr

Enter Word: xyz

Enter Meaning: oop

Enter Word: klm

Enter Meaning: ijk
Ascending order
 abc : pqr
 klm : ijk
 xyz : oop

Descending order:
 xyz : oop
 klm : ijk
 abc : pqr

Enter word to search: hds

Not found word

...Program finished with exit code 0
Press ENTER to exit console.
```

```
/*            Assignment No 6: There are flight paths between cities. If there is a flight between city A
and city B then there is an edge between the cities. The cost of the edge can be the time that flight take
to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node
can be represented by airport name or name of the city. Use adjacency list representation of the graph
or use adjacency matrix representation of the graph.  Check whether the graph is connected or not.
Justify the storage representation used.*/

#include<iostream>
#include<stdlib.h>
#include<string.h> using
namespace std; struct node {
string vertex;     int time;
   node *next;
};
class adjmatlist
{    int m[10][10],n,i,j; char ch;  string v[20];   node *head[20];  node *temp=NULL;

    public:
    adjmatlist()
    {      for(i=0;i<20;i++)
          {    head[i]=NULL;  }
    }
    void getgraph();      void
adjlist();

    void displaym();      void
displaya();
};
void adjmatlist::getgraph()
{
  cout<<"\n Enter no. of cities(max. 20): ";    cin>>n;
  cout<<"\n Enter name of cities: ";
for(i=0;i<n;i++)      cin>>v[i];
for(i=0;i<n;i++)
  {
     for(j=0;j<n;j++)
     {  cout<<"\n If path is present between city "<<v[i]<<" and "<<v[j]<<" then press enter y otherwise n: ";
cin>>ch;          if(ch=='y')
       {
        cout<<"\n Enter time required to reach city "<<v[j]<<" from "<<v[i]<<" in minutes: ";
        cin>>m[i][j];
       }
       else if(ch=='n')         {
m[i][j]=0;  }
       else
       { cout<<"\n Unknown entry";  }
     }
  }            adjlist();

}     void     adjmatlist::adjlist()     {
cout<<"\n                        ****";
for(i=0;i<n;i++)
     {  node *p=new(struct node);           p-
>next=NULL;           p->vertex=v[i];
       head[i]=p;      cout<<"\n"<<head[i]->vertex;
     }

     for(i=0;i<n;i++)       {
for(j=0;j<n;j++)
       {
              if(m[i][j]!=0)
              {
```

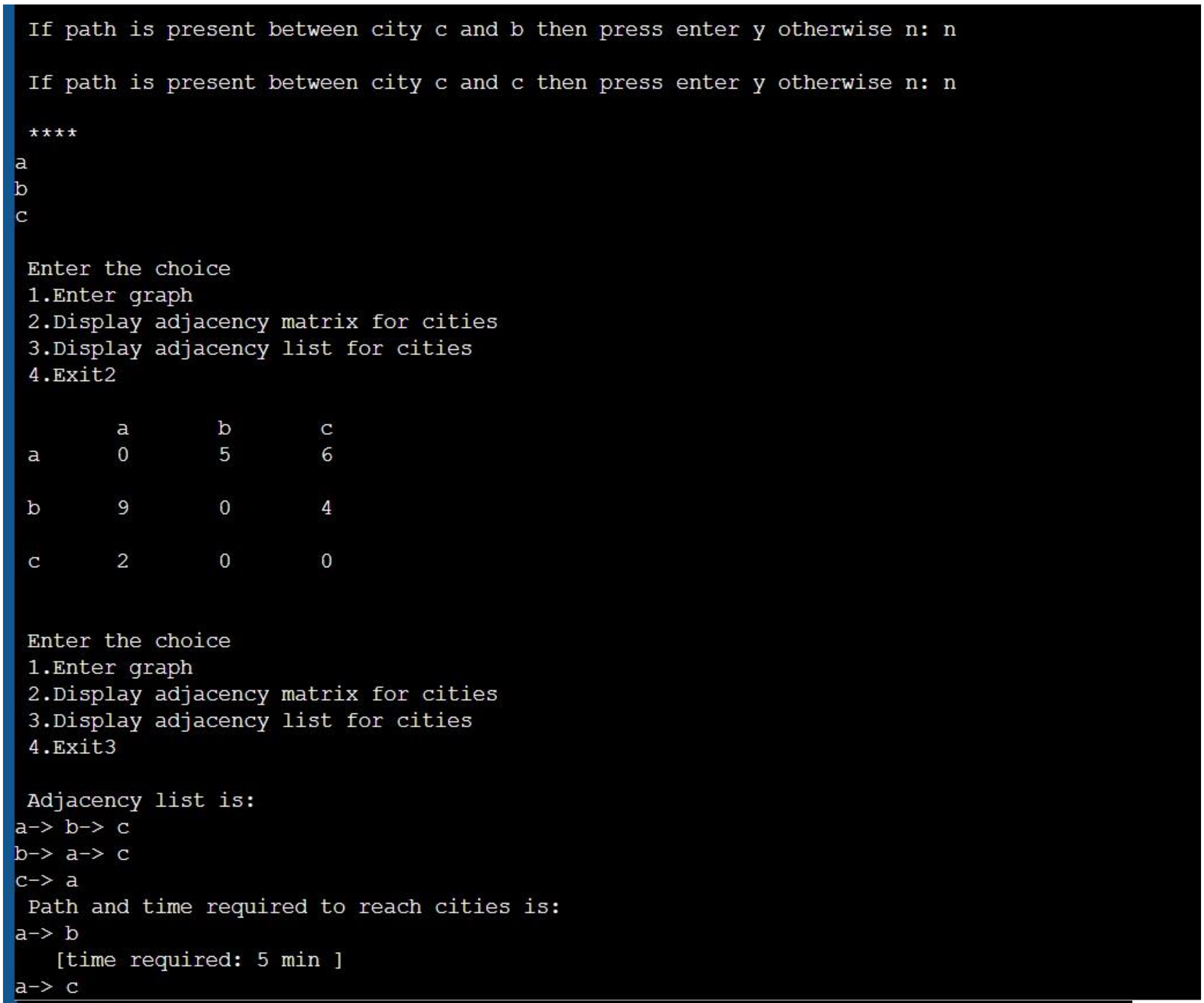
```
                        node *p=new(struct node);
                        p->vertex=v[j];                         p->time=m[i][j];
p->next=NULL;
                        if(head[i]->next==NULL)                         {
head[i]->next=p;   }
                        else
                        {  temp=head[i];
                        while(temp->next!=NULL)                         {
temp=temp->next;  }
                            temp->next=p;
                        }

                  }

          }
       }

}
void adjmatlist::displaym() {
cout<<"\n";      for(j=0;j<n;j++)      {
cout<<"\t"<<v[j];  }

     for(i=0;i<n;i++)             {     cout<<"\n
"<<v[i];             for(j=0;j<n;j++)            {
cout<<"\t"<<m[i][j];
        }
           cout<<"\n";
     }
}
void adjmatlist::displaya()
{
       cout<<"\n Adjacency list is: ";

       for(i=0;i<n;i++)
       {


                        if(head[i]==NULL)
                        {   cout<<"\n Adjacency list not present ";  break;   }                         else
                        {
                           cout<<"\n"<<head[i]->vertex;
temp=head[i]->next;                         while(temp!=NULL)
{  cout<<"-> "<<temp->vertex;
                           temp=temp->next;  }

                        }





}

         cout<<"\n Path and time required to reach cities is: ";

       for(i=0;i<n;i++)
       {


                        if(head[i]==NULL)
                        {   cout<<"\n Adjacency list not present";  break;   }                         else                         {
```

```
                        temp=head[i]->next;                        while(temp!=NULL)
                        {  cout<<"\n"<<head[i]->vertex;
                           cout<<"-> "<<temp->vertex<<"\n   [time required: "<<temp-
>time<<" min ]";
                           temp=temp->next;  }

                        }




       } } int main() {
int m;       adjmatlist
a;

   while(1)
   {
   cout<<"\n\n Enter the choice";    cout<<"\n 1.Enter graph";    cout<<"\n
2.Display adjacency matrix for cities";    cout<<"\n 3.Display adjacency
list for cities";    cout<<"\n 4.Exit";    cin>>m;

        switch(m)
       {              case 1: a.getgraph();
break;                      case 2: a.displaym();
break;

                          case 3: a.displaya();
                                  break;
case 4: exit(0);

                           default:  cout<<"\n Unknown choice";
         }     }
return 0; }
```

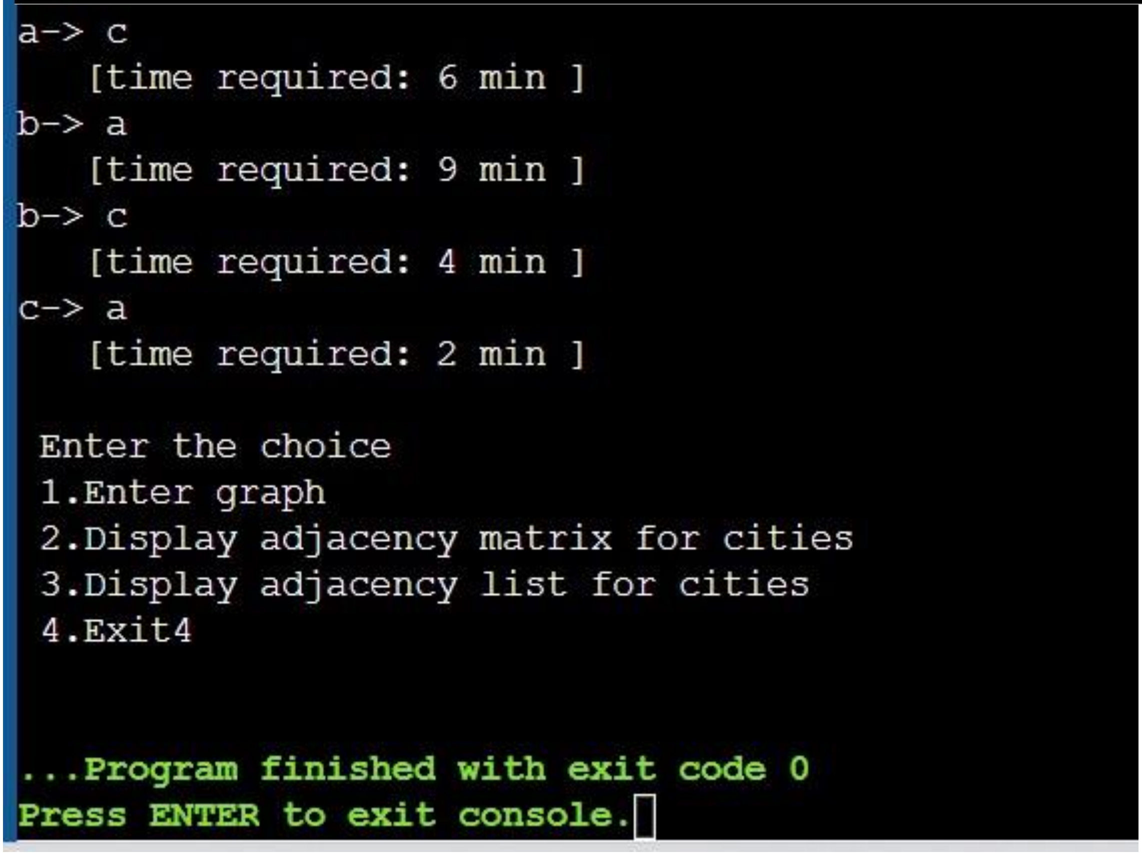
OUTPUT:



```
 Enter the choice
 1.Enter graph
 2.Display adjacency matrix for cities
 3.Display adjacency list for cities
 4.Exit 1

 Enter no. of cities(max. 20): 3

 Enter name of cities:
a
b
c

 If path is present between city a and a then press enter y otherwise n: n

 If path is present between city a and b then press enter y otherwise n: y

 Enter time required to reach city b from a in minutes: 5

 If path is present between city a and c then press enter y otherwise n: y

 Enter time required to reach city c from a in minutes: 6

 If path is present between city b and a then press enter y otherwise n: y

 Enter time required to reach city a from b in minutes: 9

 If path is present between city b and b then press enter y otherwise n: n

 If path is present between city b and c then press enter y otherwise n: y

 Enter time required to reach city c from b in minutes: 4

 If path is present between city c and a then press enter y otherwise n: y

 Enter time required to reach city a from c in minutes: 2
```



```
 If path is present between city c and b then press enter y otherwise n: n

 If path is present between city c and c then press enter y otherwise n: n

 ****
a
b
c

 Enter the choice
 1.Enter graph
 2.Display adjacency matrix for cities
 3.Display adjacency list for cities
 4.Exit2

        a       b       c
 a      0       5       6

 b      9       0       4

 c      2       0       0


 Enter the choice
 1.Enter graph
 2.Display adjacency matrix for cities
 3.Display adjacency list for cities
 4.Exit3

 Adjacency list is:
a-> b-> c
b-> a-> c
c-> a
 Path and time required to reach cities is:
a-> b
   [time required: 5 min ]
a-> c
```



```
a-> c
   [time required: 6 min ]
b-> a
   [time required: 9 min ]
b-> c
   [time required: 4 min ]
c-> a
   [time required: 2 min ]

 Enter the choice
 1.Enter graph
 2.Display adjacency matrix for cities
 3.Display adjacency list for cities
 4.Exit4


...Program finished with exit code 0
Press ENTER to exit console.
```

/* **Assignment No 7**

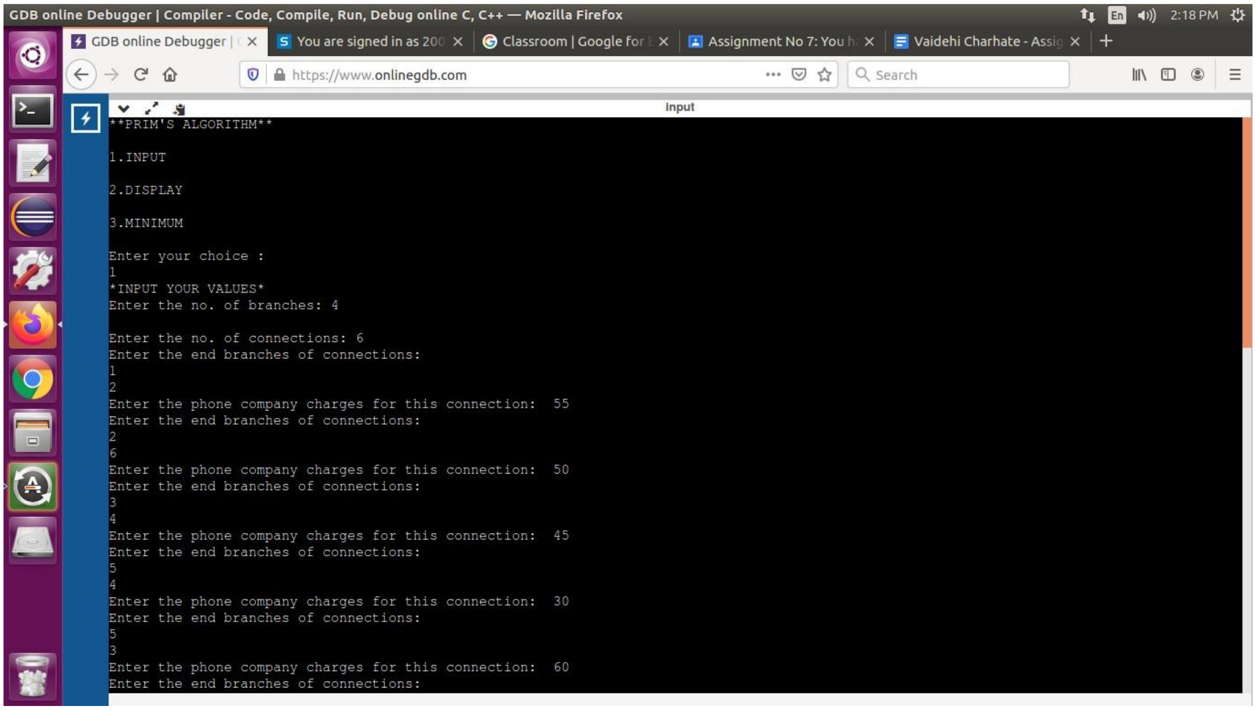
You have a business with several offices; You want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with a minimum total cost. Solve the problem by suggesting appropriate data structures. */

```
#include<iostream> using
namespace std;

class tree
{
            int a[20][20],l,u,w,i,j,v,e,visited[20];
public:
        void input();
void display();     void
minimum();
};

void tree::input()
{
            cout<<"Enter the no. of branches: ";
         cin>>v;

          for(i=0;i<v;i++)
        {
        visited[i]=0;
for(j=0;j<v;j++)
        {
         a[i][j]=999;
        }
        }

            cout<<"\nEnter the no. of connections: ";
         cin>>e;
```

```
          for(i=0;i<e;i++)
        {
   cout<<"Enter the end branches of connections:  "<<endl;     cin>>l>>u;     cout<<"Enter
the phone company charges for this connection:  ";     cin>>w;
           a[l-1][u-1]=a[u-1][l-1]=w;
        }
}

void tree::display()
{
        cout<<"\nAdjacency matrix:";              for(i=0;i<v;i++)
        {
         cout<<endl;
          for(j=0;j<v;j++)
        {
           cout<<a[i][j]<<"   ";
        }
cout<<endl;
        }
}

void tree::minimum()
{
        int p=0,q=0,total=0,min;        visited[0]=1;
             for(int count=0;count<(v-1);count++)
        {
         min=999;
          for(i=0;i<v;i++)
        {
        if(visited[i]==1)
{            for(j=0;j<v;j++)
      {
        if(visited[j]!=1)
{                           if(min > a[i][j])
{                   min=a[i][j];
p=i;                q=j;
}
                    }
       }
          }
          }
visited[p]=1;
visited[q]=1;
total=total+min;
cout<<"Minimum cost
connection is"<<(p+1)<<" -
> "<<(q+1)<<"  with charge
: "<<min<< endl;

          }
                   cout<<"The minimum total cost of connections of all branches is: "<<total<<endl;
}

int main()
{
          int ch;
tree t;
          do
          {
          cout<<"**PRIM'S ALGORITHM**"<<endl;        cout<<"\n1.INPUT\n
\n2.DISPLAY\n \n3.MINIMUM\n"<<endl;        cout<<"Enter your choice :"<<endl;
cin>>ch;

           switch(ch)
          {
               case 1: cout<<"*INPUT YOUR VALUES*"<<endl;
          t.input();
break;
```

```
              case 2: cout<<"*DISPLAY THE CONTENTS*"<<endl;

          t.display();

         break;


             case 3: cout<<"*MINIMUM*"<<endl;

         t.minimum();

break;

         }


         }while(ch!=4);

return 0;

}
```

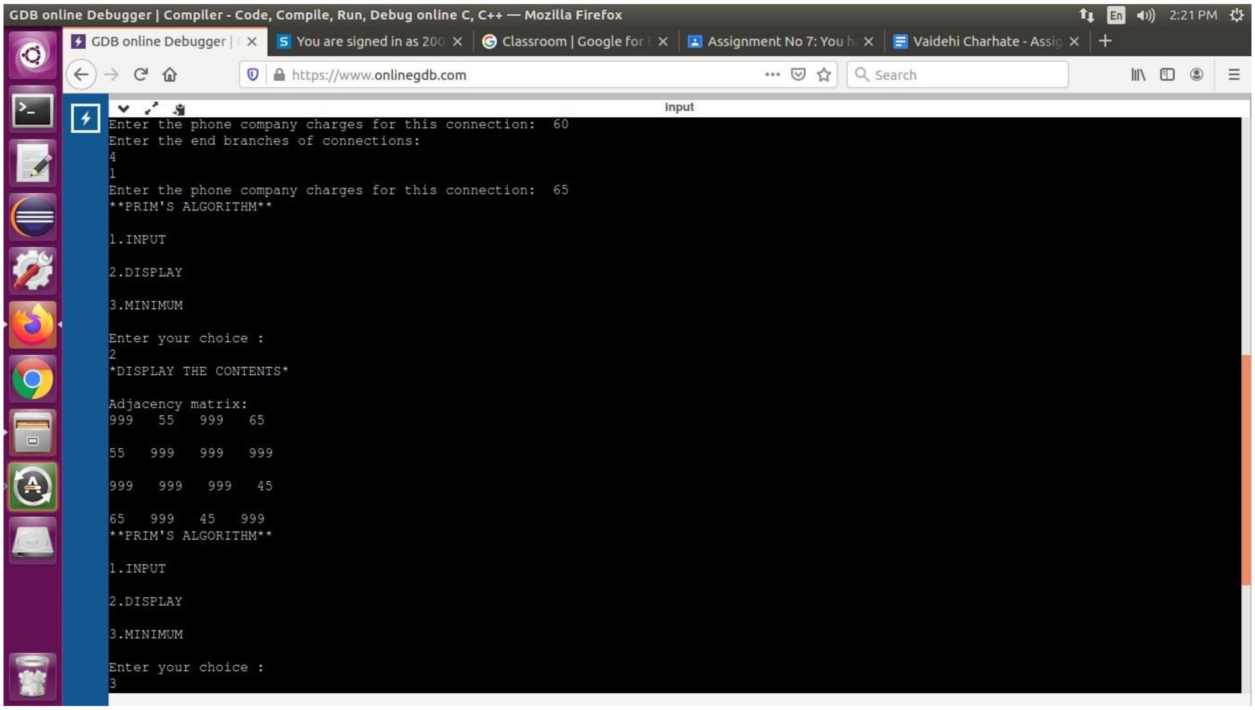
OUTPUT:



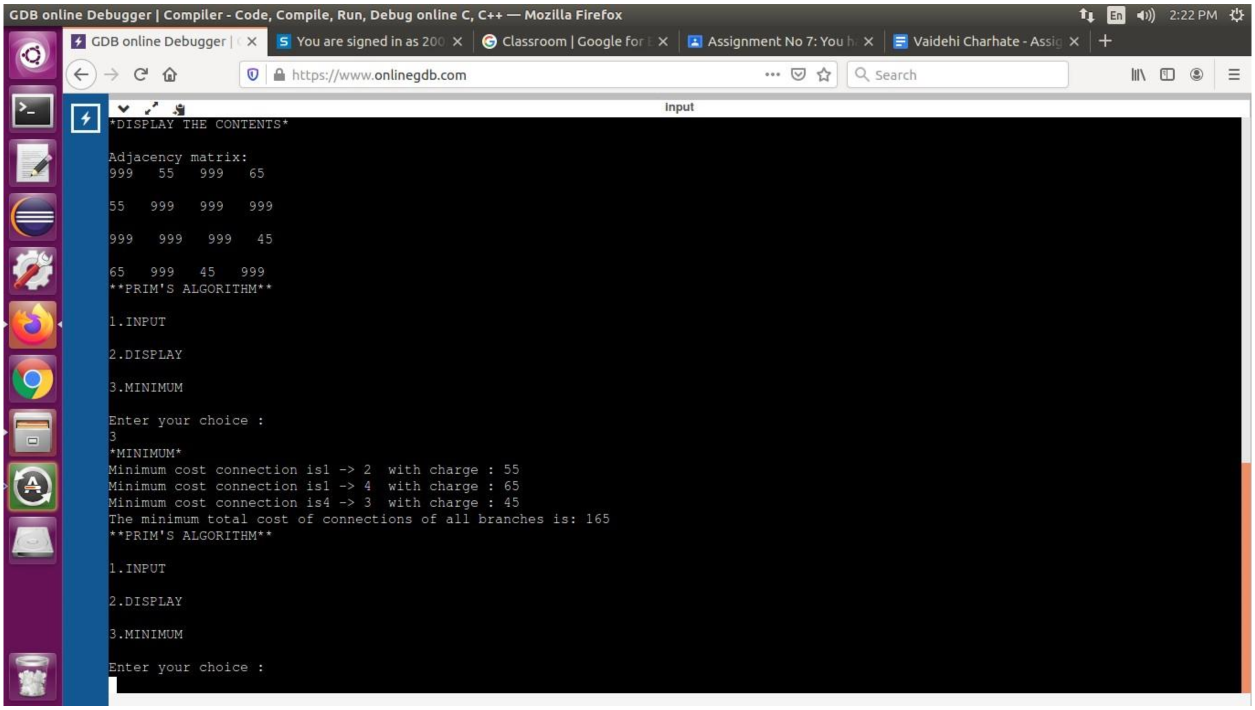
GDB online Debugger | Compiler - Code, Compile, Run, Debug online C, C++ — Mozilla Firefox

https://www.onlinegdb.com

```
**PRIM'S ALGORITHM**

1.INPUT

2.DISPLAY

3.MINIMUM

Enter your choice :
1
*INPUT YOUR VALUES*
Enter the no. of branches: 4

Enter the no. of connections: 6
Enter the end branches of connections:
1
2
Enter the phone company charges for this connection:  55
Enter the end branches of connections:
2
6
Enter the phone company charges for this connection:  50
Enter the end branches of connections:
3
4
Enter the phone company charges for this connection:  45
Enter the end branches of connections:
5
4
Enter the phone company charges for this connection:  30
Enter the end branches of connections:
5
3
Enter the phone company charges for this connection:  60
Enter the end branches of connections:
```



```
Enter the phone company charges for this connection:  60
Enter the end branches of connections:
4
1
Enter the phone company charges for this connection:  65
**PRIM'S ALGORITHM**

1.INPUT

2.DISPLAY

3.MINIMUM

Enter your choice :
2
*DISPLAY THE CONTENTS*

Adjacency matrix:
999   55   999   65

55   999   999   999

999   999   999   45

65   999   45   999
**PRIM'S ALGORITHM**

1.INPUT

2.DISPLAY

3.MINIMUM

Enter your choice :
3
```



```
*DISPLAY THE CONTENTS*

Adjacency matrix:
999   55   999   65

55   999   999   999

999   999   999   45

65   999   45   999
**PRIM'S ALGORITHM**

1.INPUT

2.DISPLAY

3.MINIMUM

Enter your choice :
3
*MINIMUM*
Minimum cost connection is1 -> 2  with charge : 55
Minimum cost connection is1 -> 4  with charge : 65
Minimum cost connection is4 -> 3  with charge : 45
The minimum total cost of connections of all branches is: 165
**PRIM'S ALGORITHM**

1.INPUT

2.DISPLAY

3.MINIMUM

Enter your choice :
```

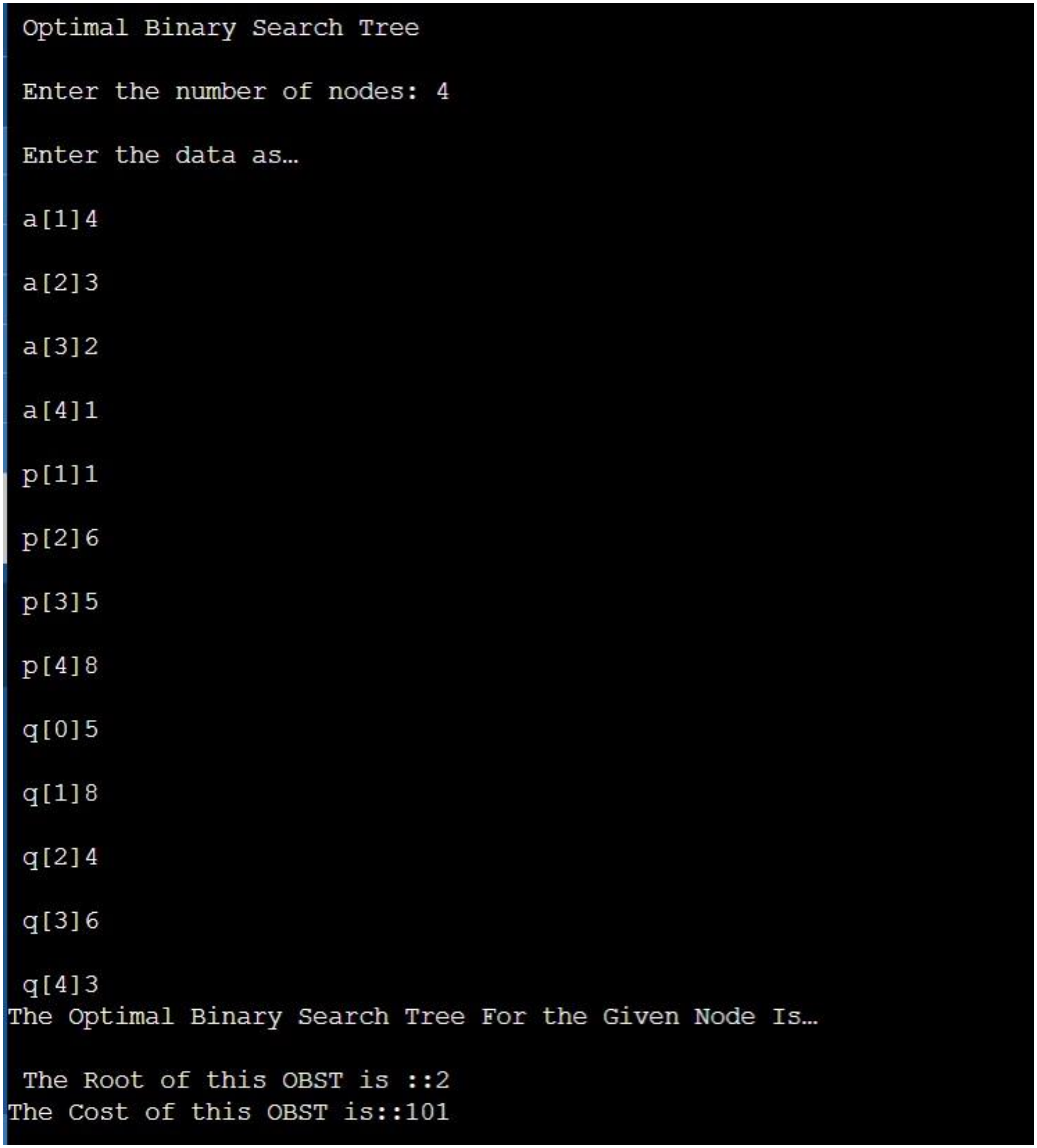
/*ASSIGNMENT NO 08: Given sequence k = k1 <k2 < ... <kn of n sorted keys, with a search probability pi for each key ki . Build the Binary search tree that has the least search cost given the access probability for each key.*/

CODE:

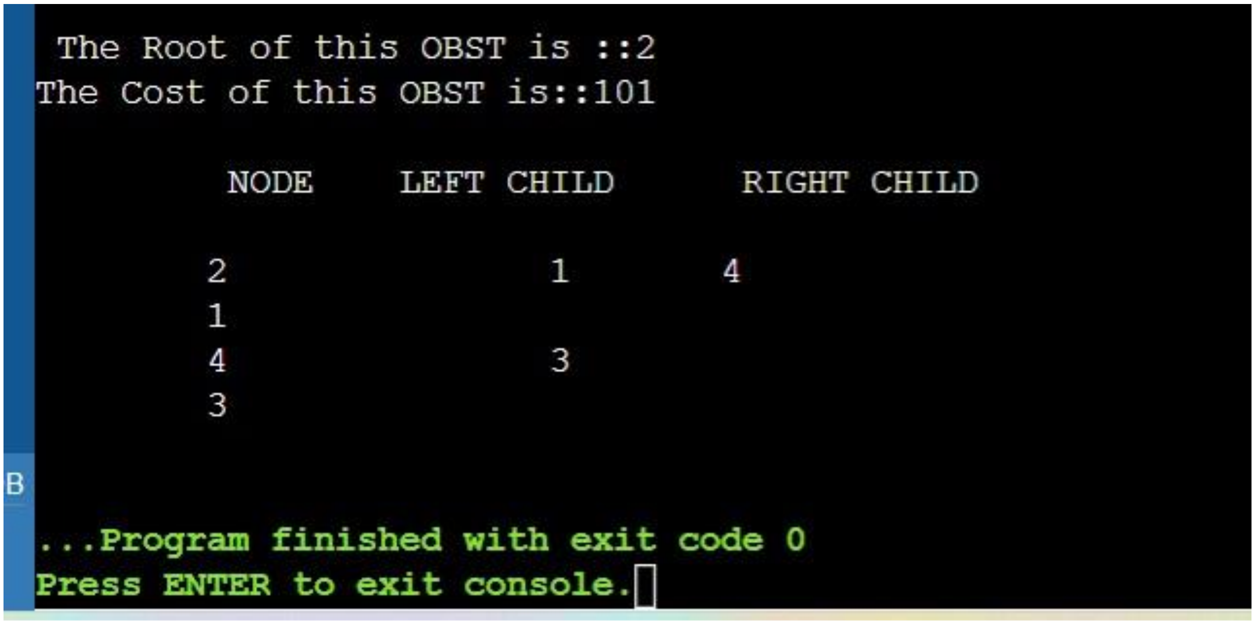
```
/* This program is to implement optimal binary search tree*/
#include<iostream>                using
namespace  std;  #define  SIZE  10
class OBST
{ int p[SIZE]; // Probabilities with which we search for an element int q[SIZE];//Probabilities that an
element is not found int a[SIZE];//Elements from which OBST is to be built int w[SIZE][SIZE];//Weight
'w[i][j]' of a tree having root
//'r[i][j]' int c[SIZE][SIZE];//Cost 'c[i][j] of a tree having root 'r[i][j] int r[SIZE][SIZE];//represents
root int n; // number of nodes public:
/* This function accepts the input data */ void get_data()
{
int i;
cout<<"\n Optimal Binary Search Tree \n";
cout<<"\n Enter the number of nodes: "; cin>>n;
cout<<"\n Enter the data as…\n"; for(i=1;i<=n;i++)
{ cout<<"\n a["<<i<<"]"; cin>>a[i]; }
for(i=1;i<=n;i++)
{ cout<<"\n p["<<i<<"]"; cin>>p[i]; }
for(i=0;i<=n;i++)
{ cout<<"\n q["<<i<<"]"; cin>>q[i];
}
}
/* This function returns a value in the range 'r[i][j-1]' to 'r[i+1][j]'so that the cost 'c[i][k-1]+c[k][j]'is minimum */
int Min_Value(int i,int j)
{ int m,k; int
minimum=3
2000;
for(m=r[i][j-
1];m<=r[i+1
][j];m++)
{ if((c[i][m-1]+c[m][j])<minimum)
{  minimum=c[i][m-1]+c[m][j];  k=m;
} } return k;
}
/* This function builds the table from all the given probabilities It basically computes C,r,W values */ void
build_OBST()
{  int i,j,k,l,m;
for(i=0;i<n;i++)
 {
//initialize w[i][i]=q[i];
r[i][i]=c[i][i]=0;
//Optimal trees with one node w[i][i+1]=q[i]+q[i+1]+p[i+1];
r[i][i+1]=i+1; c[i][i+1]=q[i]+q[i+1]+p[i+1];

 }  w[n][n]=q[n];
r[n][n]=c[n][n]=0;
 //Find optimal trees with ‘m’ nodes  for(m=2;m<=n;m++)
 {
for(i=0;i<=n-m;i++)
{  j=i+m;  w[i][j]=w[i][j-1]+p[j]+q[j];
k=Min_Value(i,j);  c[i][j]=w[i][j]+c[i][k-1]+c[k][j];
r[i][j]=k;
}
 }
}
/* This function builds the tree from the tables made by the OBST function */ void build_tree()
{  int i,j,k;
 int queue[20],front=-1,rear=-1;

 cout<<"The Optimal Binary Search Tree For the Given Node Is…\n";

 cout<<"\n The Root of this OBST is ::"<<r[0][n];
 cout<<"\nThe Cost of this OBST is::"<<c[0][n];  cout<<"\n\n\t NODE \t LEFT CHILD \t RIGHT
CHILD ";  cout<<"\n";  queue[++rear]=0;  queue[++rear]=n;  while(front!=rear)
 {
i=queue[++front]; j=queue[++front];
k=r[i][j];  cout<<"\n\t"<<k; if(r[i][k-1]!=0)
{ cout<<"\t\t"<<r[i][k-1];
queue[++rear]=i; queue[++rear]=k-1;
} else cout<<"\t\t"; if(r[k][j]!=0)
{ cout<<"\t"<<r[k][j];
queue[++rear]=k;
queue[++rear]=j;
} else cout<<"\t";
 }  cout<<"\n"; }
};
```

```
int main() {
OBST obj;
obj.get_data(); obj.build_OBST();
obj.build_tree(); return 0;
}
```

OUTPUT:



```
 Optimal Binary Search Tree

 Enter the number of nodes: 4

 Enter the data as…

 a[1]4

 a[2]3

 a[3]2

 a[4]1

 p[1]1

 p[2]6

 p[3]5

 p[4]8

 q[0]5

 q[1]8

 q[2]4

 q[3]6

 q[4]3
The Optimal Binary Search Tree For the Given Node Is…

 The Root of this OBST is ::2
The Cost of this OBST is::101
```



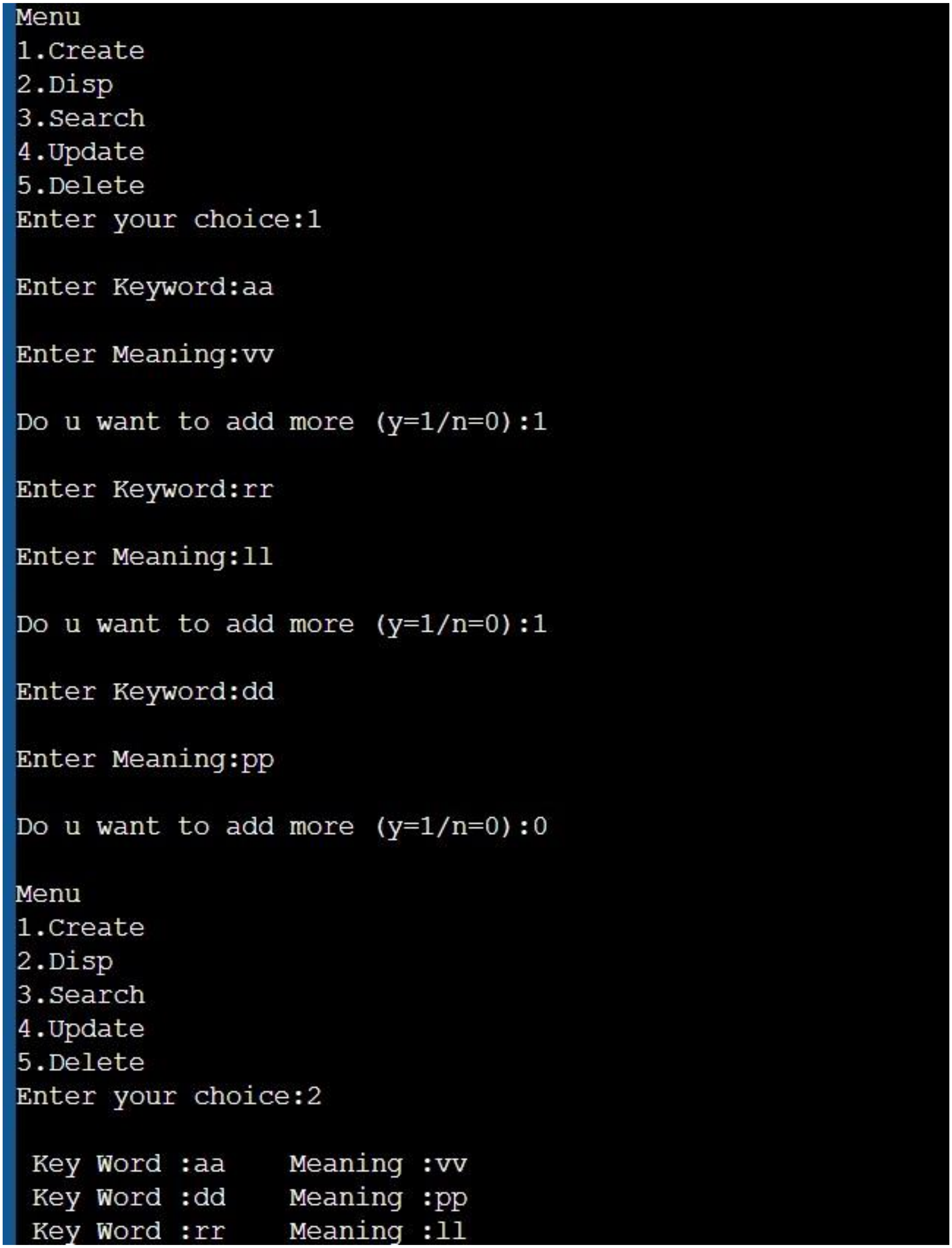
```
 The Root of this OBST is ::2
The Cost of this OBST is::101

         NODE    LEFT CHILD      RIGHT CHILD

        2               1       4
        1
        4               3
        3


...Program finished with exit code 0
Press ENTER to exit console.
```

```
/*Assignment No 9

A Dictionary stores keywords & its meanings.

Provide facility for adding new keywords, deleting keywords, updating values of any entry.

Provide facility to display whole data sorted in ascending/ Descending order.

Also find how many maximum comparisons may require for finding any keyword.

Use Binary Search Tree for implementation.*/


#include"iostream"
#include<string.h> using
namespace std;


typedef struct node
{
 char    k[20];        char
m[20];      class    node
*left;     class   node   *
right;
}node;


class dict
{
public:
 node *root;  void create();  void disp(node *);
void insert(node * root,node *temp);  int
search(node *,char []);  int update(node
*,char []);

 node* del(node *,char []);  node *
min(node *);
};


void dict :: create()
{
 class node *temp;
 int ch;
 do
 {
  temp          =          new          node;
cout<<"\nEnter              Keyword:";
cin>>temp->k;        cout<<"\nEnter
Meaning:";   cin>>temp->m;

  temp->left = NULL;   temp->right =
NULL;

  if(root == NULL)
  {
   root = temp;
  }
  else
  {
   insert(root, temp);
  }
  cout<<"\nDo u want to add more (y=1/n=0):"; cin>>ch;
 }
 while(ch == 1);

}

void dict ::  insert(node * root,node *temp)
{
 if(strcmp (temp->k, root->k) < 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);
 }

}

void dict:: disp(node * root)
{
 if( root != NULL)
 {
  disp(root->left);
  cout<<"\n Key Word :"<<root->k;   cout<<"\t
Meaning :"<<root->m;
  disp(root->right);
 }
}

int dict :: search(node * root,char k[20])
{
 int c=0;
 while(root != NULL)
 {
  c++;   if(strcmp (k,root->k) == 0)
  {
   cout<<"\nNo of Comparisons:"<<c;
   return 1;
  }
  if(strcmp (k, root->k) < 0)    root =
root->left;   if(strcmp (k, root->k) >
0)    root = root->right;
 }

 return -1;
```

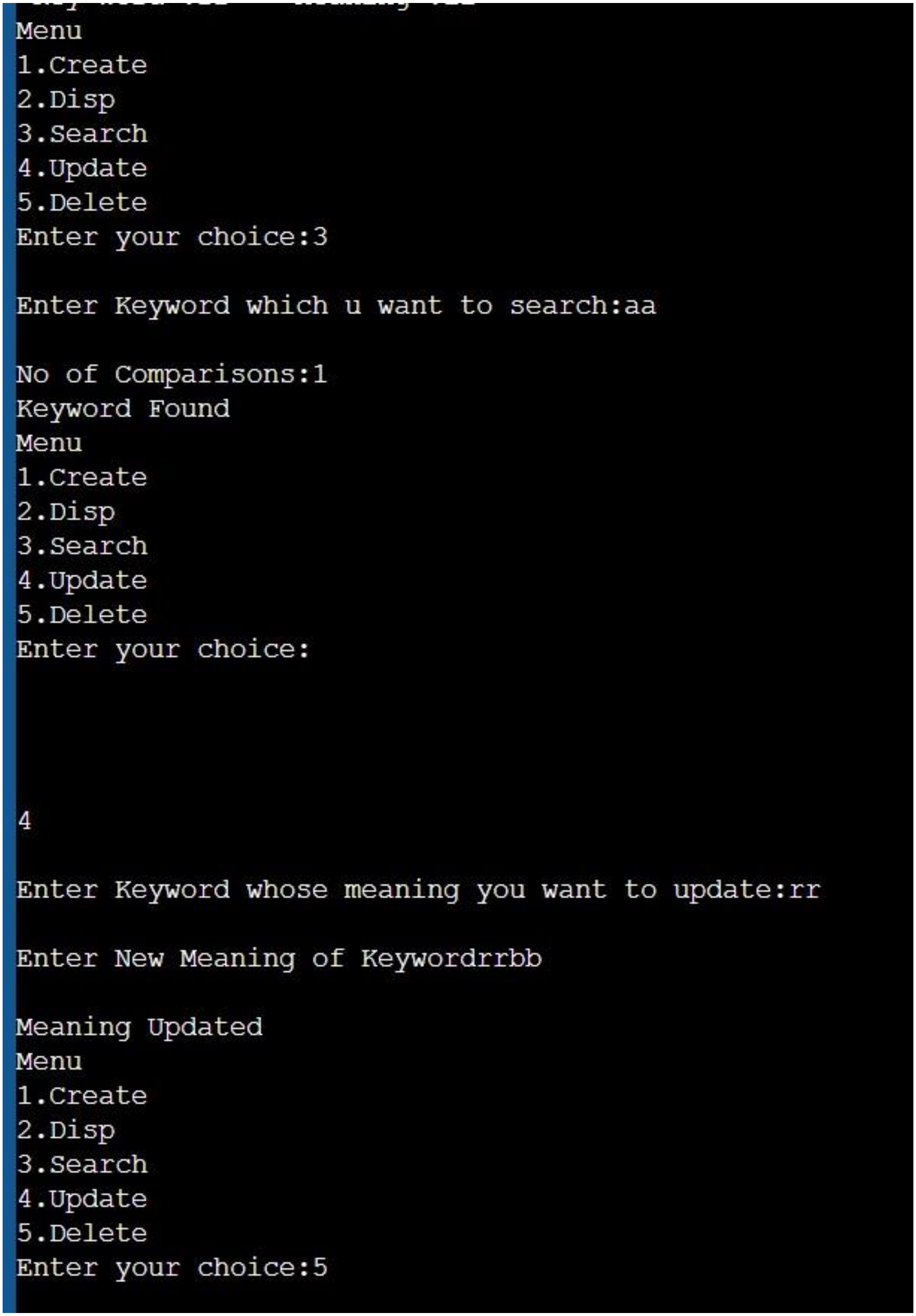
```
}
int dict :: update(node * root,char k[20])
{
 while(root != NULL)
 {
   if(strcmp (k,root->k) == 0)
  {
   cout<<"\nEnter New Meaning of Keyword"<<root->k;    cin>>root-
>m;    return 1;
  }
  if(strcmp (k, root->k) < 0)    root =
root->left;   if(strcmp (k, root->k) >
0)    root = root->right;
 }
 return -1;
}
node* dict :: del(node * root,char k[20])
{
 node *temp;

 if(root == NULL)
 {
  cout<<"\nElement Not Found";
  return root;
 }

 if (strcmp(k,root->k) < 0)
 {
  root->left = del(root->left, k);   return root;
 }
 if (strcmp(k,root->k) > 0)
 {
   root->right = del(root->right, k);    return root;
 }
 if (root->right==NULL&&root->left==NULL)
 {
  temp = root;   delete
temp;   return NULL;
  }
  if(root->right==NULL)
  {
  temp = root;   root =
root->left;   delete
temp;   return root;
  }
  else if(root->left==NULL)
  {
  temp = root;   root =
root->right;   delete
temp;   return root;
  }
  temp = min(root->right);   strcpy(root->k,temp-
>k);   root->right = del(root->right, temp->k);
return root;


}


node * dict :: min(node *q)
{
 while(q->left != NULL)
 {
  q = q->left;
 }
 return q;
}

int main()
```

```
{
 int ch;  dict
d;
 d.root = NULL;

 do
 {
  cout<<"\nMenu\n1.Create\n2.Disp\n3.Search\n4.Update\n5.Delete\nEnter your choice:";   cin>>ch;

  switch(ch)
  {
case 1: d.create();
  break; case 2: if(d.root ==
NULL)
  {
  cout<<"\nNo such Keyword";
  }
  else
  {
  d.disp(d.root);
  }
  break; case 3: if(d.root ==
NULL)
 {
  cout<<"\nDictionary is Empty. First add keywords then try again ";
 }
  else
 {

       cout<<"\nEnter Keyword which u want to search:";   char k[20];
cin>>k;

  if( d.search(d.root,k) == 1)   cout<<"\nKeyword Found";
  else
  cout<<"\nKeyword Not Found";
 }
  break; case 4:
  if(d.root == NULL)
  {
  cout<<"\nDictionary is Empty. First add keywords then try again ";
 }
  else
  {
  cout<<"\nEnter Keyword whose meaning you want to update:";
  char k[20];   cin>>k;
if(d.update(d.root,k) == 1)
  cout<<"\nMeaning Updated";
  else
  cout<<"\nMeaning Not Found";
  }
  break; case 5:
if(d.root == NULL)
  {
  cout<<"\nDictionary is Empty. First add keywords then try again ";
  }
  else
  {
  cout<<"\nEnter Keyword which u want to delete:";   char
k[20];   cin>>k;   if(d.root == NULL)
  {
  cout<<"\nNo any Keyword";
  }
  else
  {
  d.root = d.del(d.root,k);
    }
  }
  }
 }
 while(ch<=5);  return 0;

}
```

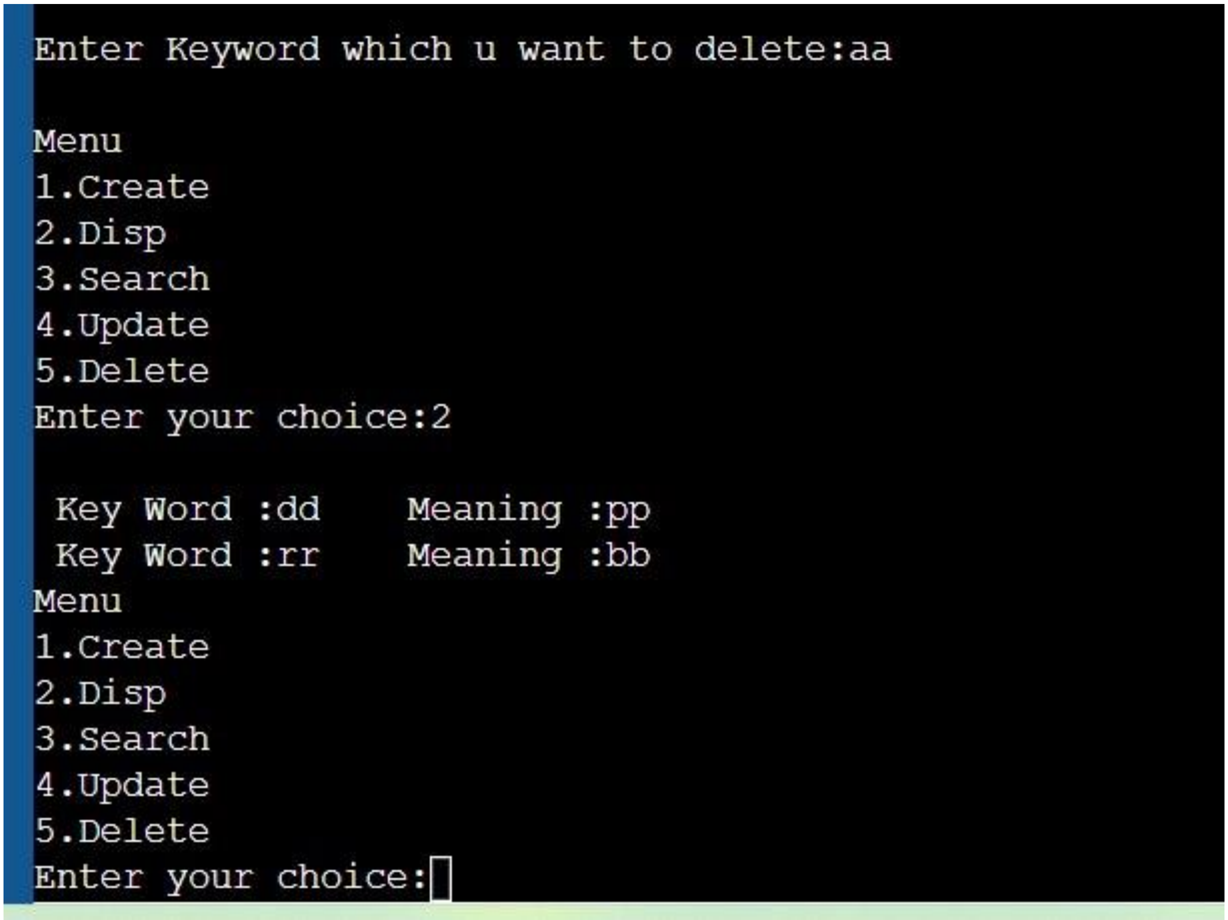
OUTPUT:



```
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter your choice:1

Enter Keyword:aa

Enter Meaning:vv

Do u want to add more (y=1/n=0):1

Enter Keyword:rr

Enter Meaning:ll

Do u want to add more (y=1/n=0):1

Enter Keyword:dd

Enter Meaning:pp

Do u want to add more (y=1/n=0):0

Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter your choice:2

 Key Word :aa    Meaning :vv
 Key Word :dd    Meaning :pp
 Key Word :rr    Meaning :ll
```



```
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter your choice:3

Enter Keyword which u want to search:aa

No of Comparisons:1
Keyword Found
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter your choice:





4

Enter Keyword whose meaning you want to update:rr

Enter New Meaning of Keywordrrbb

Meaning Updated
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter your choice:5
```



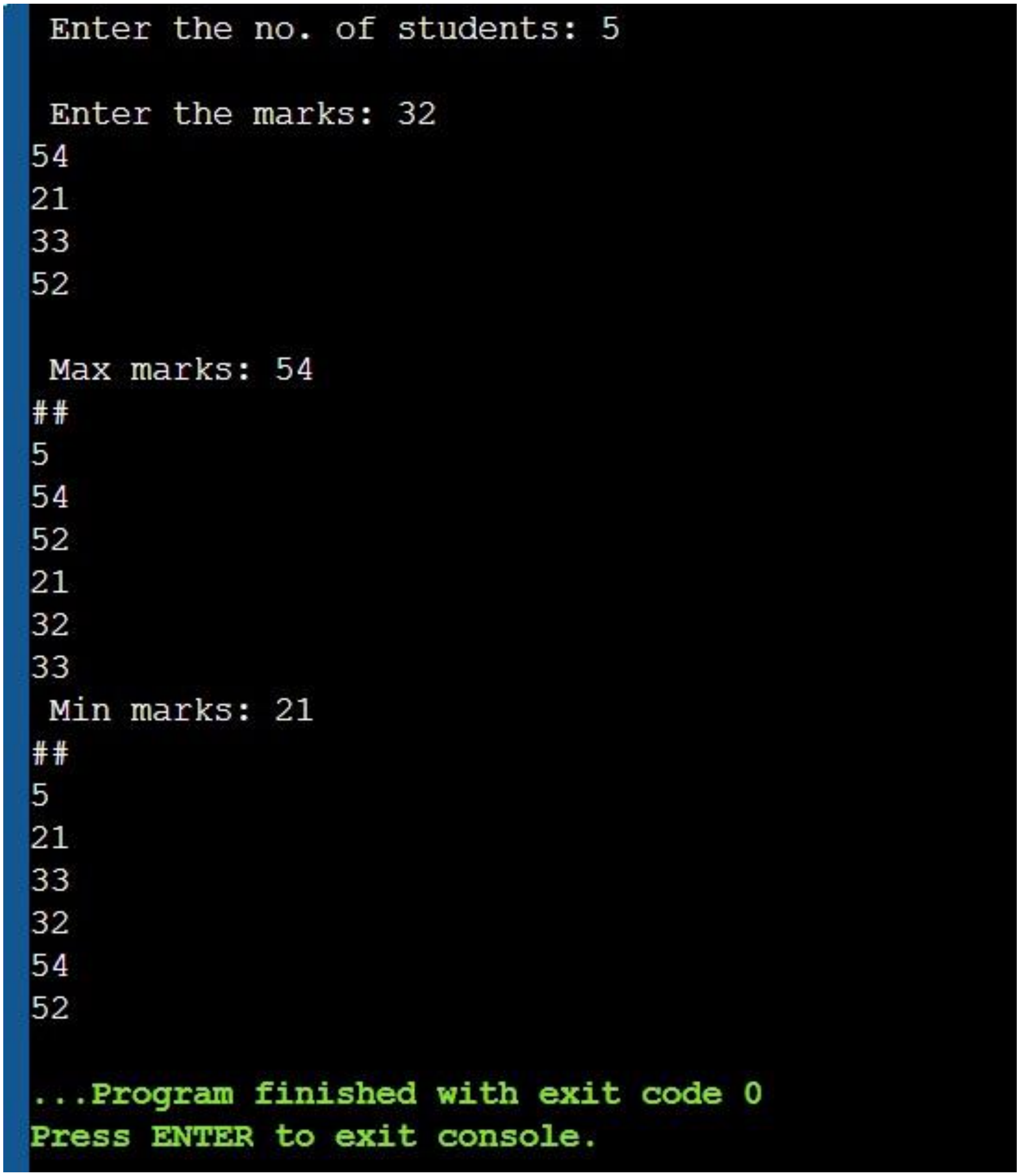
```
Enter Keyword which u want to delete:aa

Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter your choice:2

 Key Word :dd    Meaning :pp
 Key Word :rr    Meaning :bb
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter your choice:
```

```
/*Assignment No 10:   Heap data structure, Read the marks obtained by students of second year, Find out maximum and
minimum marks obtained

*/


#include<iostream> using

namespace std;


class hp

{

   int heap[20],heap1[20],x,n1,i;

public:    hp()

   { heap[0]=0;  heap1[0]=0;

   }

   void getdata();    void insert1(int

heap[],int);    void upadjust1(int

heap[],int);    void insert2(int

heap1[],int);    void upadjust2(int

heap1[],int);    void minmax();

};

void hp::getdata()

{

   cout<<"\n enter the no. of students";

cin>>n1;    cout<<"\n enter the marks";

   for(i=0;i<n1;i++)

   {   cin>>x;

       insert1(heap,x);        insert2(heap1,x);

   }

}

void hp::insert1(int heap[20],int x)

{    int n;

n=heap[0];

heap[n+1]=x;

heap[0]=n+1;


   upadjust1(heap,n+1);
```

```
}
void hp::upadjust1(int heap[20],int i)
{
   int temp;
   while(i>1&&heap[i]>heap[i/2])
   {
     temp=heap[i];       heap[i]=heap[i/2];
heap[i/2]=temp;
     i=i/2;
   }
}
void hp::insert2(int heap1[20],int x)
{    int n;
  n=heap1[0];    heap1[n+1]=x;
heap1[0]=n+1;


  upadjust2(heap1,n+1);
}
void hp::upadjust2(int heap1[20],int i)
{
   int temp1;     while(i>1&&heap1[i]<heap1[i/2])
   {
     temp1=heap1[i];       heap1[i]=heap1[i/2];
heap1[i/2]=temp1;
     i=i/2;
   }
}
void hp::minmax()
{
  cout<<"\n max marks"<<heap[1];
cout<<"\n##";    for(i=0;i<=n1;i++)    {
cout<<"\n"<<heap[i];  }    cout<<"\n min
marks"<<heap1[1];    cout<<"\n##";
for(i=0;i<=n1;i++)
```

```
   {   cout<<"\n"<<heap1[i];  }
}
int main()
{
  hp h;
  h.getdata();
  h.minmax();
  return 0;
}
```

OUTPUT:

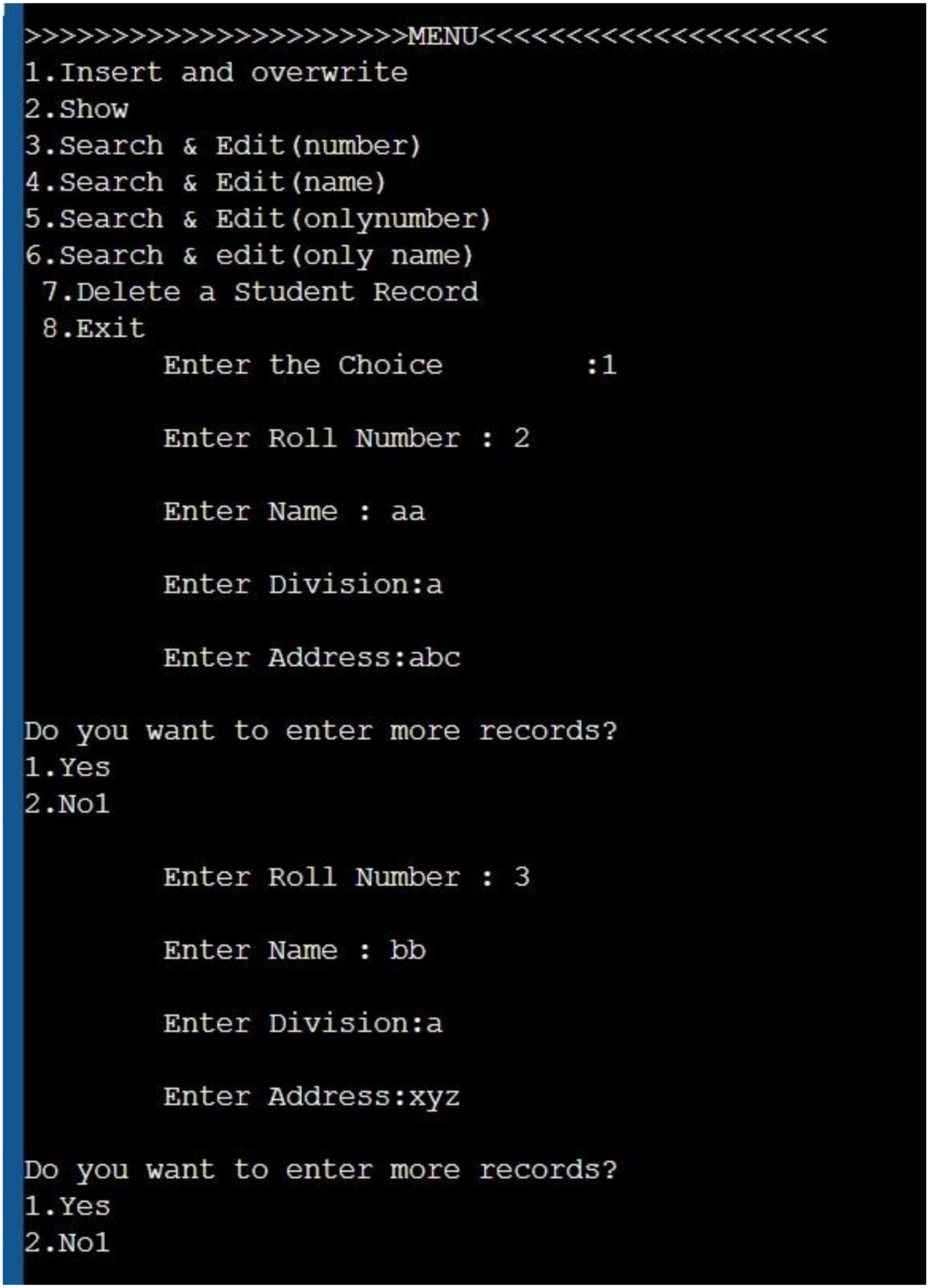


```
 Enter the no. of students: 5

 Enter the marks: 32
54
21
33
52

 Max marks: 54
##
5
54
52
21
32
33
 Min marks: 21
##
5
21
33
32
54
52

...Program finished with exit code 0
Press ENTER to exit console.
```

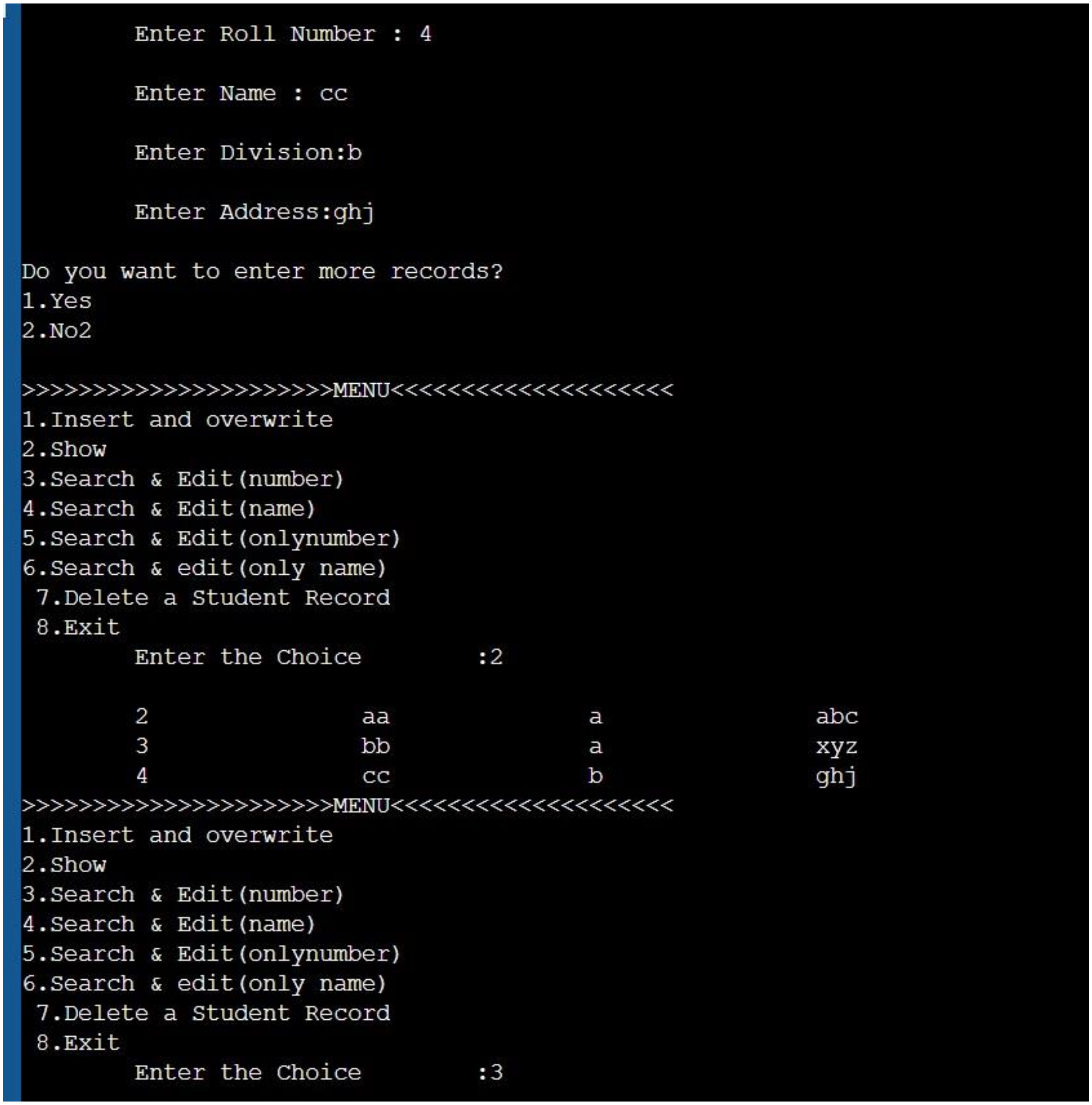
```
/*Assignment No 11: Department maintains a student information. The file contains roll number, name,
division and address. Allow user to add, delete information of student. Display information of particular employee. If
record of student does not exist an appropriate message is displayed. If it is, then the system displays the student
details. Use sequential file to main the data.
*/

/*Department maintains a student information.
The file contains roll number, name, division and address.
Allow user to add, delete information of student.
Display information of particular employee.
If record of student does not exist an appropriate message is displayed.
If it is, then the system displays the student details. Use sequential file to main the data.*/

#include<iostream>
#include<fstream>
#include<cstring> using
namespace std;
class tel
 {

 public:
 int rollNo,roll1;  char
name[10];
 char div;  char
address[20];  void
accept()
 {
  cout<<"\n\tEnter Roll Number : ";
cin>>rollNo;   cout<<"\n\tEnter Name : ";
cin>>name;   cout<<"\n\tEnter Division:";
cin>>div;   cout<<"\n\tEnter Address:";
cin>>address;
 }
       void accept2()
       {
            cout<<"\n\tEnter the Roll No. to modify : ";               cin>>rollNo;

       }

       void accept3()

       {

            cout<<"\n\tEnter the name to modify : ";              cin>>name;

       }

       int getRollNo()

       {

        return rollNo;

       }

  void show()

  {


  cout<<"\n\t"<<rollNo<<"\t\t"<<name<<"\t\t"<<div<<"\t\t"<<address;

  }

};

int main()

{

 int i,n,ch,ch1,rec,start,count,add,n1,add2,start2,n2,y,a,b,on,oname,add3,start3,n3,y1,add4,start4,n4;  char

name[20],name2[20];

 tel t1;

 count=0;  fstream

g,f;  do

 {

  cout<<"\n>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<";

  cout<<"\n1.Insert and overwrite\n2.Show\n3.Search & Edit(number)\n4.Search &
Edit(name)\n5.Search & Edit(onlynumber)\n6.Search & edit(only name)\n 7.Delete a Student Record\n
8.Exit\n\tEnter the Choice\t:";

cin>>ch;   switch(ch)

  {

  case 1:

   f.open("StuRecord.txt",ios::out);    x:t1.accept();

   f.write((char*) &t1,(sizeof(t1)));    cout<<"\nDo you want to enter more

records?\n1.Yes\n2.No";
```

```
  cin>>ch1;
if(ch1==1)
goto x;     else
   {
    f.close();      break;
   }

 case 2:
  f.open("StuRecord.txt",ios::in);
  f.read((char*) &t1,(sizeof(t1)));
  //cout<<"\n\tRoll No.\t\tName \t\t Division \t\t Address";    while(f)
  {
   t1.show();
   f.read((char*) &t1,(sizeof(t1)));
  }
  f.close();
break;   case 3:
  cout<<"\nEnter the roll number you want to find";    cin>>rec;
  f.open("StuRecord.txt",ios::in|ios::out);
  f.read((char*)&t1,(sizeof(t1)));    while(f)
  {
   if(rec==t1.rollNo)
   {
    cout<<"\nRecord found";
    add=f.tellg();
    f.seekg(0,ios::beg);           start=f.tellg();
n1=(add-start)/(sizeof(t1));
    f.seekp((n1-1)*sizeof(t1),ios::beg);      t1.accept();
    f.write((char*) &t1,(sizeof(t1)));
    f.close();
count++;      break;
   }
   f.read((char*)&t1,(sizeof(t1)));
      }
  if(count==0)          cout<<"\nRecord not
found";
  f.close();    break;

 case 4:
   cout<<"\nEnter the name you want to find and edit";     cin>>name;
  f.open("StuRecord.txt",ios::in|ios::out);
  f.read((char*)&t1,(sizeof(t1)));    while(f)
  {
   y=(strcmp(name,t1.name));
   if(y==0)
   {
    cout<<"\nName found";
    add2=f.tellg();
    f.seekg(0,ios::beg);      start2=f.tellg();
n2=(add2-start2)/(sizeof(t1));
    f.seekp((n2-1)*sizeof(t1),ios::beg);      t1.accept();
    f.write((char*) &t1,(sizeof(t1)));
    f.close();      break;
   }
        f.read((char*)&t1,(sizeof(t1)));
  }
  break;
case 5:
         cout<<"\n\tEnter the roll number you want to modify";
         cin>>on;
         f.open("StuRecord.txt",ios::in|ios::out);
         f.read((char*) &t1,(sizeof(t1)));            while(f)
         {
           if(on==t1.rollNo)
           {
             cout<<"\n\tNumber found";
             add3=f.tellg();
        f.seekg(0,ios::beg);         start3=f.tellg();
n3=(add3-start3)/(sizeof(t1));
        f.seekp((n3-1)*(sizeof(t1)),ios::beg);        t1.accept2();
        f.write((char*)&t1,(sizeof(t1)));
        f.close();
break;
       }
       f.read((char*)&t1,(sizeof(t1)));
     }
     break;
   case 6:
      cout<<"\nEnter the name you want to find and edit";   cin>>name2;
 f.open("StuRecord.txt",ios::in|ios::out);
 f.read((char*)&t1,(sizeof(t1)));   while(f)
 {
  y1=(strcmp(name2,t1.name));
  if(y1==0)
  {
   cout<<"\nName found";
   add4=f.tellg();
   f.seekg(0,ios::beg);    start4=f.tellg();
n4=(add4-start4)/(sizeof(t1));
   f.seekp((n4-1)*sizeof(t1),ios::beg);    t1.accept3();
   f.write((char*) &t1,(sizeof(t1)));
   f.close();    break;
  }
     f.read((char*)&t1,(sizeof(t1)));
 }
 break;
case 7:    int
roll;
   cout<<"Please Enter the Roll No. of Student Whose Info You Want to Delete: ";
   cin>>roll;
   f.open("StuRecord.txt",ios::in);
   g.open("temp.txt",ios::out);
   f.read((char *)&t1,sizeof(t1));     while(!f.eof())
   {
     if (t1.getRollNo() != roll)
       g.write((char *)&t1,sizeof(t1));
      f.read((char *)&t1,sizeof(t1));
   }
  cout << "The record with the roll no. " << roll << " has been deleted " << endl;     f.close();
   g.close();     remove("StuRecord.txt");
rename("temp.txt","StuRecord.txt");
    break;    case
8:
    cout<<"\n\tThank you";
    break;



     }
 }while(ch!=8);
}
```

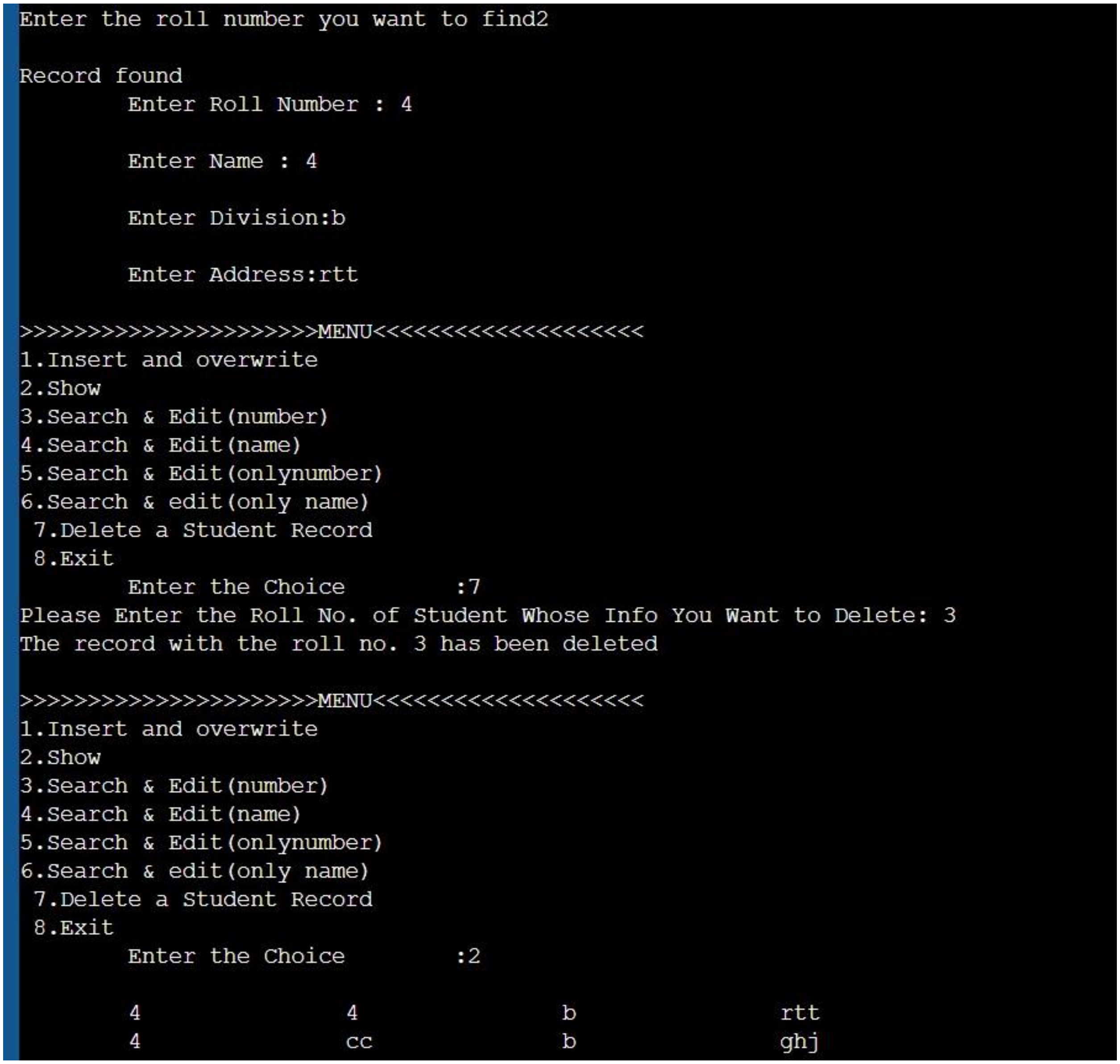
OUTPUT:



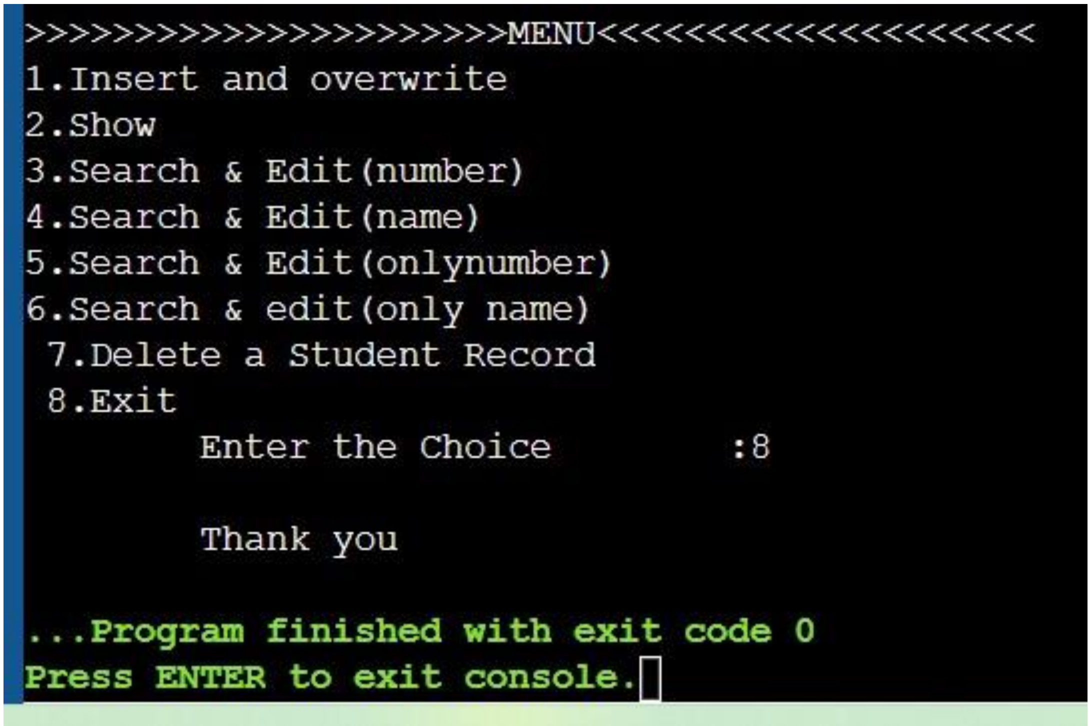
```
>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<
1.Insert and overwrite
2.Show
3.Search & Edit(number)
4.Search & Edit(name)
5.Search & Edit(onlynumber)
6.Search & edit(only name)
 7.Delete a Student Record
 8.Exit
        Enter the Choice        :1

        Enter Roll Number : 2

        Enter Name : aa

        Enter Division:a

        Enter Address:abc

Do you want to enter more records?
1.Yes
2.No1

        Enter Roll Number : 3

        Enter Name : bb

        Enter Division:a

        Enter Address:xyz

Do you want to enter more records?
1.Yes
2.No1
```



```
        Enter Roll Number : 4

        Enter Name : cc

        Enter Division:b

        Enter Address:ghj

Do you want to enter more records?
1.Yes
2.No2

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<
1.Insert and overwrite
2.Show
3.Search & Edit(number)
4.Search & Edit(name)
5.Search & Edit(onlynumber)
6.Search & edit(only name)
 7.Delete a Student Record
 8.Exit
        Enter the Choice        :2

        2               aa              a               abc
        3               bb              a               xyz
        4               cc              b               ghj
>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<
1.Insert and overwrite
2.Show
3.Search & Edit(number)
4.Search & Edit(name)
5.Search & Edit(onlynumber)
6.Search & edit(only name)
 7.Delete a Student Record
 8.Exit
        Enter the Choice        :3
```



```
Enter the roll number you want to find2

Record found
        Enter Roll Number : 4

        Enter Name : 4

        Enter Division:b

        Enter Address:rtt

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<
1.Insert and overwrite
2.Show
3.Search & Edit(number)
4.Search & Edit(name)
5.Search & Edit(onlynumber)
6.Search & edit(only name)
 7.Delete a Student Record
 8.Exit
        Enter the Choice        :7
Please Enter the Roll No. of Student Whose Info You Want to Delete: 3
The record with the roll no. 3 has been deleted

>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<
1.Insert and overwrite
2.Show
3.Search & Edit(number)
4.Search & Edit(name)
5.Search & Edit(onlynumber)
6.Search & edit(only name)
 7.Delete a Student Record
 8.Exit
        Enter the Choice        :2

        4               4               b               rtt
        4               cc              b               ghj
```



```
>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<
1.Insert and overwrite
2.Show
3.Search & Edit(number)
4.Search & Edit(name)
5.Search & Edit(onlynumber)
6.Search & edit(only name)
 7.Delete a Student Record
 8.Exit
        Enter the Choice        :8

        Thank you

...Program finished with exit code 0
Press ENTER to exit console.
```

```
/*Assignment  no 12: Company maintains employee information as employee ID, name, designation and salary. Allow
user to add, delete information of employee. Display information of particular employee. If employee does not exist
an appropriate message is displayed. If it is, then the system displays the employee details. Use index sequential file
to maintain the data
*/

#include<iostream>
#include<fstream>
#include<stdio.h>

using namespace std;

//Employee class Declaration class
Employee{
   private:
int code;
      char      name[20];
float salary;    public:
      void read();      void
display();
      //will return employee code      int
getEmpCode()        { return code;}      //will
return employee salary      int getSalary()        {
return salary;}      //will update employee salary
void updateSalary(float s)  { salary=s;} };


//Read employee record void
Employee::read(){    cout<<"Enter
employee code: ";    cin>>code;
cout<<"Enter name: ";    cin.ignore(1);
cin.getline(name,20);    cout<<"Enter
salary: ";    cin>>salary;
}

//Display employee record void
Employee::display()
```

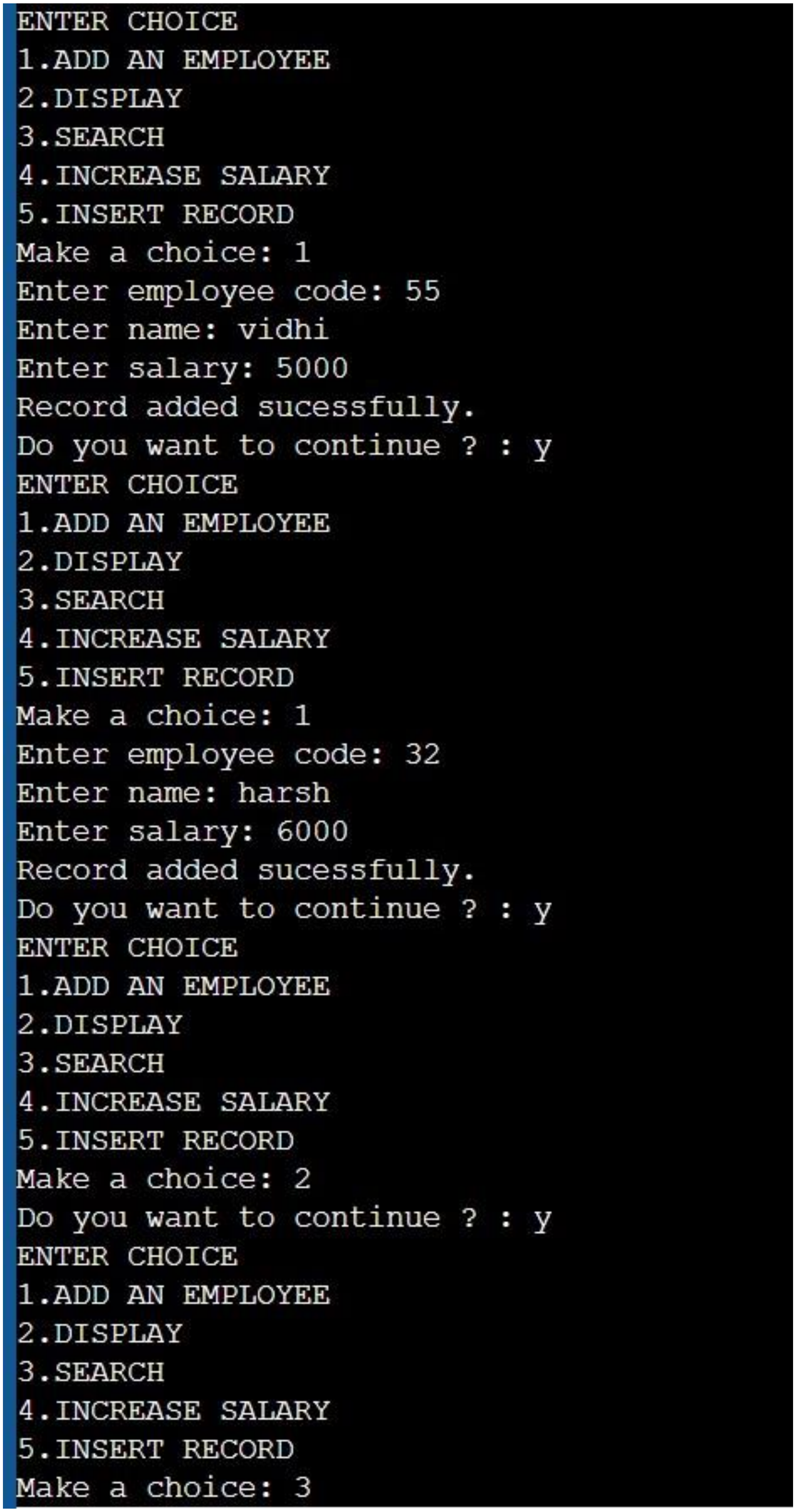
```
{
   cout<<code<<" "<<name<<"\t"<<salary<<endl;
}

//global declaration fstream file;

//Will delete file when program is being executed //because
we are create file in append mode void deleteExistingFile(){
remove("EMPLOYEE.DAT");
}

//function to append record into file
void appendToFille(){
   Employee    x;

   //Read employee record from user
   x.read();

   file.open("EMPLOYEE.DAT",ios::binary|ios::app);
   if(!file){
      cout<<"ERROR IN CREATING FILE\n";
      return;
   }
   //write into file     file.write((char*)&x,sizeof(x));
   file.close();
   cout<<"Record added sucessfully.\n";
}

void displayAll(){
   Employee    x;

   file.open("EMPLOYEE.DAT",ios::binary|ios::in);
   if(!file){
      cout<<"ERROR IN OPENING FILE \n";
      return;
```

```
    }
    while(file){     if(file.read((char*)&x,sizeof(x)))
if(x.getSalary()>=10000 && x.getSalary()<=20000)
            x.display();
    }
  file.close();
}

void searchForRecord(){     //read
employee id     Employee    x;
    int c;
    int isFound=0;

    cout<<"Enter employee code: ";
    cin>>c;


    file.open("EMPLOYEE.DAT",ios::binary|ios::in);
    if(!file){
        cout<<"ERROR IN OPENING FILE \n";
        return;
    }
    while(file){        if(file.read((char*)&x,sizeof(x))){
if(x.getEmpCode()==c){                cout<<"RECORD
FOUND\n";
                x.display();
isFound=1;                break;
            }
        }
    }
    if(isFound==0){        cout<<"Record not
found!!!\n";
    }
    file.close();
}
```

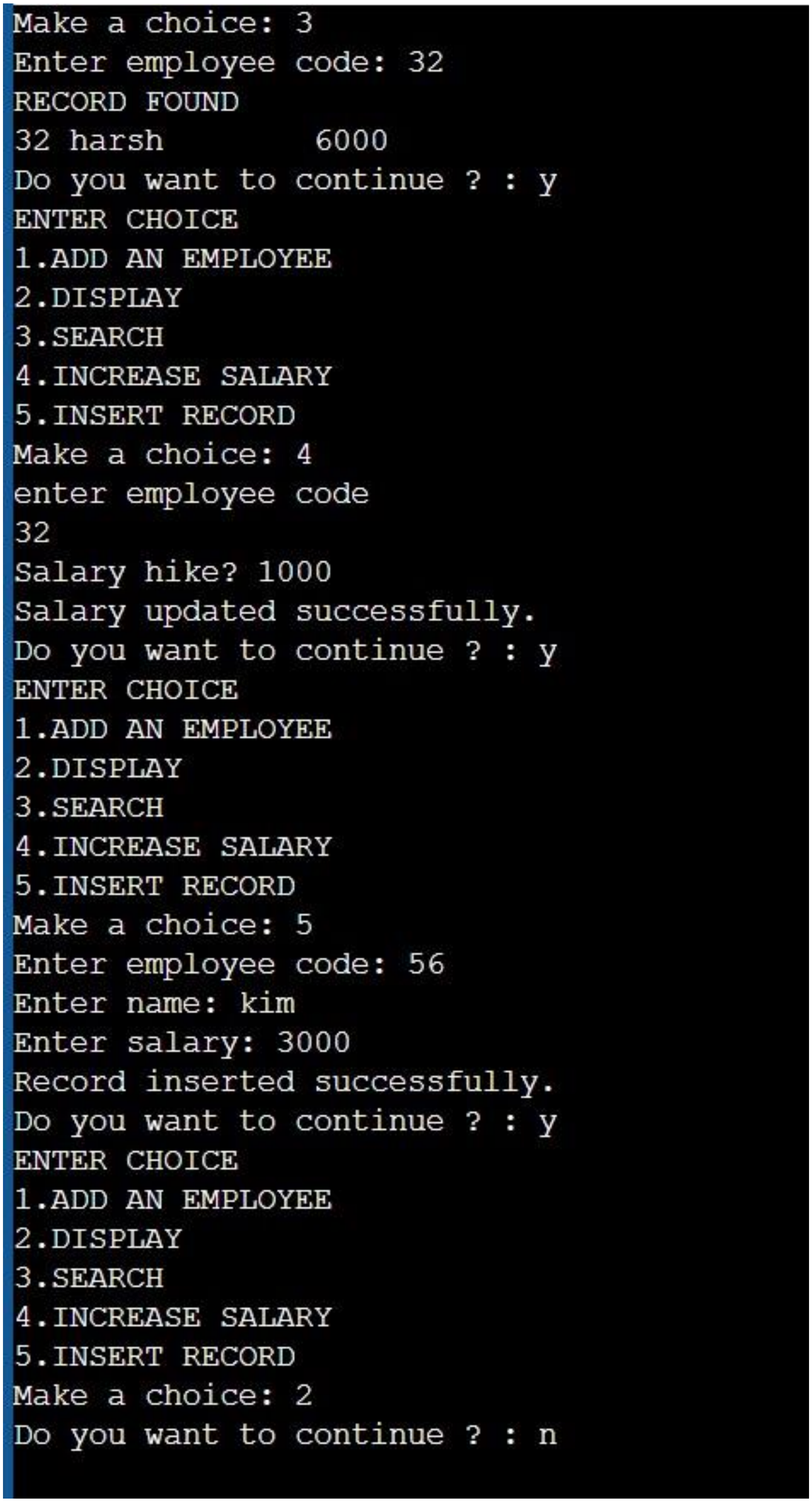
```
//Function to increase salary void
increaseSalary(){     //read
employee id     Employee    x;
   int c;
   int isFound=0;     float
sal;


   cout<<"enter employee code \n";
   cin>>c;



   file.open("EMPLOYEE.DAT",ios::binary|ios::in);
   if(!file){
      cout<<"ERROR IN OPENING FILE \n";
      return;
   }
   while(file){
if(file.read((char*)&x,sizeof(x))){
if(x.getEmpCode()==c){                cout<<"Salary
hike? ";                cin>>sal;
            x.updateSalary(x.getSalary()+sal);                isFound=1;                break;
         }
      }
   }
   if(isFound==0){        cout<<"Record not
found!!!\n";
   }
   file.close();
   cout<<"Salary updated successfully."<<endl;
}

//Insert record by assuming that records are in
//ascending order void
insertRecord(){
  //read employee record

  Employee    x;

  Employee newEmp;


  //Read record to insert     newEmp.read();


  fstream fin;     //read file in input mode
file.open("EMPLOYEE.DAT",ios::binary|ios::in);
  //open file in write mode     fin.open("TEMP.DAT",ios::binary|ios::out);
   if(!file){
cout<<"Err
or in
opening
EMPLOYEE.
DAT
file!!!\n";

    return;
  }    if(!fin){
    cout<<"Error in opening TEMP.DAT file!!!\n";
    return;
  }
  while(file){        if(file.read((char*)&x,sizeof(x))){
      if(x.getEmpCode()>newEmp.getEmpCode()){               fin.write((char*)&newEmp,
sizeof(newEmp));
      }
      //no need to use else            fin.write((char*)&x,
sizeof(x));
    }
  }


  fin.close();     file.close();


  rename("TEMP.DAT","EMPLOYEE.DAT");
remove("TEMP.DAT");     cout<<"Record inserted
successfully."<<endl;
```

```
}


int main()
{     char ch;
//if required
then only
remove the
file
deleteExistin
gFile();


    do{
int n;


    cout<<"ENTER CHOICE\n"<<"1.ADD AN
EMPLOYEE\n"<<"2.DISPLAY\n"<<"3.SEARCH\n"<<"4.INCREASE SALARY\n"<<"5.INSERT RECORD\n";
    cout<<"Make a choice: ";     cin>>n;


    switch(n){          case 1:
appendToFille();
break;          case 2 :
displayAll();           break;
case 3:
          searchForRecord();
break;        case 4:
increaseSalary();           break;
case 5:           insertRecord();
break;


        default :
             cout<<"Invalid Choice\n";
    }


    cout<<"Do you want to continue ? : ";     cin>>ch;


    }while(ch=='Y'||ch=='y');
    return 0;
}
```

OUTPUT:



```
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 1
Enter employee code: 55
Enter name: vidhi
Enter salary: 5000
Record added sucessfully.
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 1
Enter employee code: 32
Enter name: harsh
Enter salary: 6000
Record added sucessfully.
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 2
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 3
```



```
Make a choice: 3
Enter employee code: 32
RECORD FOUND
32 harsh        6000
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 4
enter employee code
32
Salary hike? 1000
Salary updated successfully.
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 5
Enter employee code: 56
Enter name: kim
Enter salary: 3000
Record inserted successfully.
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 2
Do you want to continue ? : n
```