Experiment no: 1

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

```
#include<iostream>
#include<stdlib.h>
#include<string.h>
using namespace std;
struct node
{   char name[20];
    node *next;
    node *down;
     int flag;
};
class Gll
{   char ch[20];    int n,i;
    node *head=NULL,*temp=NULL,*t1=NULL,*t2=NULL;
    public:
    node *create();
    void insertb();
    void insertc();
    void inserts();
    void insertss();
    void displayb();

};
node *Gll::create()
{
   node *p=new(struct node);
   p->next=NULL;
   p->down=NULL;
   p->flag=0;
   cout<<"\n enter the name::";
   cin>>p->name;
   return p;
}
void Gll::insertb()
{

        if(head==NULL)
          {    t1=create();
               head=t1;
          }
          else
          {
            cout<<"\n book exist \n ";
          }
}
void Gll::insertc()
{
     if(head==NULL)
```

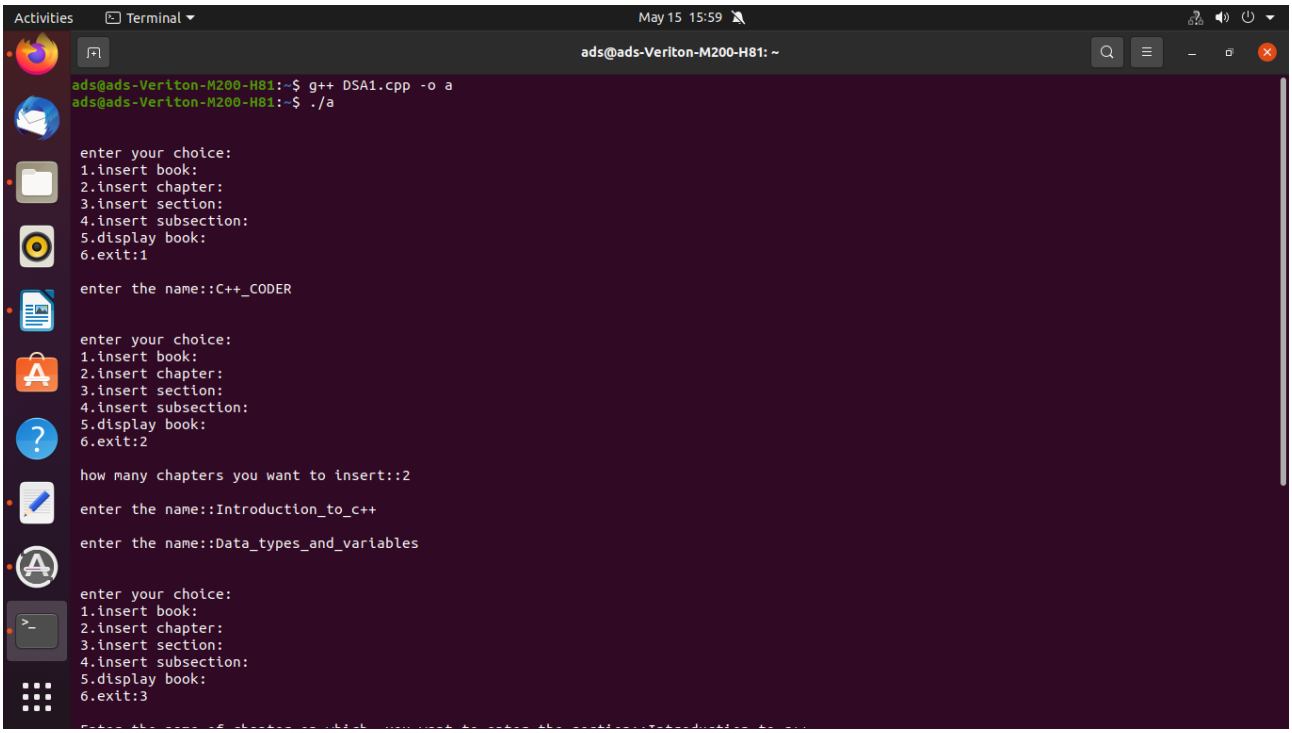
```
        {
              cout<<"\n there is no book::";
         }
         else
         {    cout<<"\n how many chapters you want to insert::";
             cin>>n;
             for(i=0;i<n;i++)
             {
             t1=create();
             if(head->flag==0)
             {  head->down=t1;   head->flag=1;    }
             else
             {   temp=head;
                 temp=temp->down;
                while(temp->next!=NULL)
                    temp=temp->next;
                 temp->next=t1;
             }
             }
        }


}
void Gll::inserts()
{

       if(head==NULL)
        {
              cout<<"\n there is no book::";
         }
         else
         {    cout<<"\n Enter the name of chapter on which  you want to enter the section::";
              cin>>ch;

              temp=head;
            if(temp->flag==0)
            {   cout<<"\n their are no chapters on in book::";
            }
            else
            {    temp=temp->down;
             while(temp!=NULL)
              {
                  if(!strcmp(ch,temp->name))
                  {
                          cout<<"\n how many sections you want to enter::";
                          cin>>n;
                          for(i=0;i<n;i++)
                          {

                                  t1=create();
                                 if(temp->flag==0)
                                 {      temp->down=t1;
```

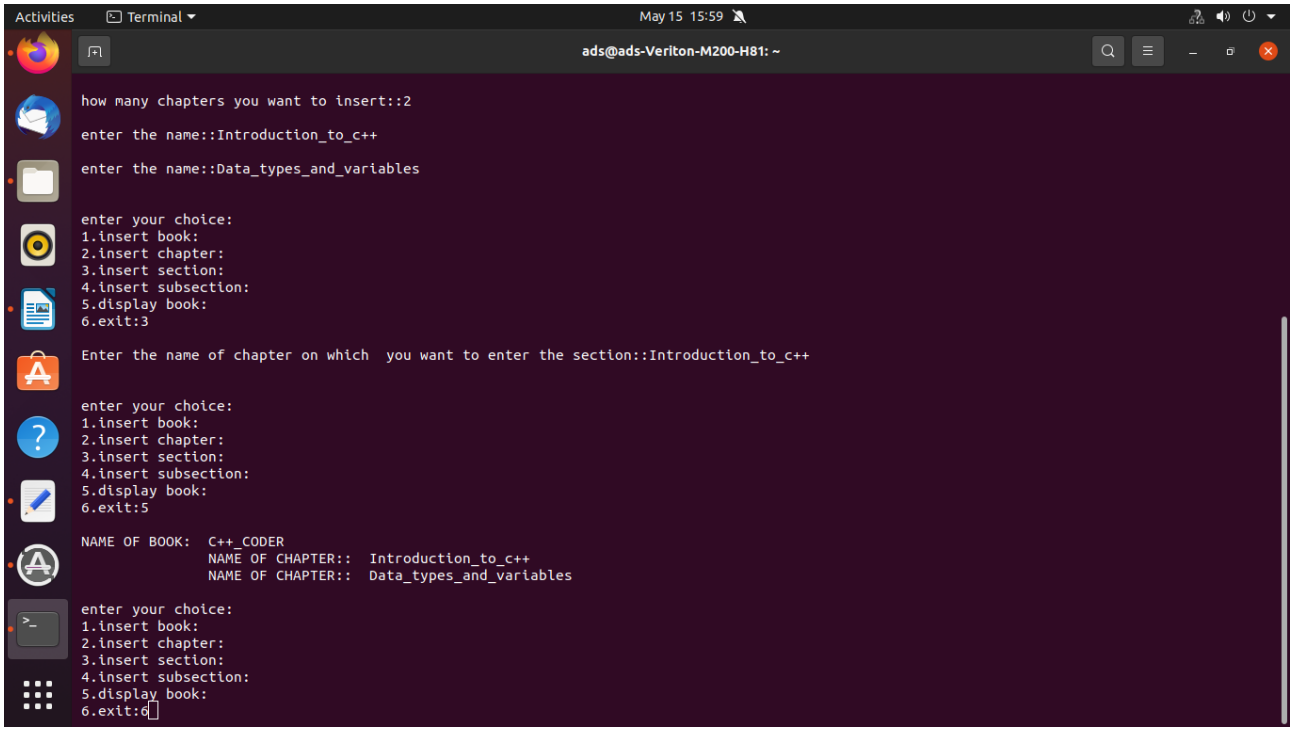
```
                                temp->flag=1;  cout<<"\n******";
                                t2=temp->down;

                             }
                            else
                             {
                                    cout<<"\n#####";
                                     while(t2->next!=NULL)
                                     {     t2=t2->next;          }
                                           t2->next=t1;
                                 }
                         }
                           break;
                    }
                        temp=temp->next;
               }
              }
        }
}
void Gll::insertss()
{

       if(head==NULL)
        {
              cout<<"\n there is no book::";
         }
         else
         {    cout<<"\n Enter the name of chapter on which  you want to enter the section::";
              cin>>ch;

              temp=head;
             if(temp->flag==0)
             {   cout<<"\n their are no chapters on in book::";
             }
             else
             {    temp=temp->down;
              while(temp!=NULL)
               {
                   if(!strcmp(ch,temp->name))
                   {
                     cout<<"\n enter name of section in which you want to enter the sub section::";
                     cin>>ch;

                    if(temp->flag==0)
                    {   cout<<"\n their are no sections ::";   }
                    else
                    {       temp=temp->down;
                           while(temp!=NULL)
                           {
                              if(!strcmp(ch,temp->name))
                              {
```

```
                    cout<<"\n how many subsections you want to enter::";
                      cin>>n;
              for(i=0;i<n;i++)
                        {

                            t1=create();
                               if(temp->flag==0)
                               {      temp->down=t1;

                                      temp->flag=1;  cout<<"\n******";
                                      t2=temp->down;

                               }
                              else
                               {
                                          cout<<"\n#####";
                                           while(t2->next!=NULL)
                                           {     t2=t2->next;          }
                                                 t2->next=t1;
                                  }
                             }
                              break;
                          }      temp=temp->next;
                        }
                    }
                  }

                        temp=temp->next;
             }
            }
       }
}
void Gll::displayb()
{

            if(head==NULL)
            {  cout<<"\n book not exist::";
            }
            else
            {
             temp=head;

               cout<<"\n NAME OF BOOK:  "<<temp->name;
                   if(temp->flag==1)
                   {
                   temp=temp->down;

                     while(temp!=NULL)
                     {     cout<<"\n\t\tNAME OF CHAPTER::  "<<temp->name;
                          t1=temp;
                          if(t1->flag==1)
                          {  t1=t1->down;
```

```
                    while(t1!=NULL)
                    {     cout<<"\n\t\t\t\tNAME OF SECTION:: "<<t1->name;
                          t2=t1;
                          if(t2->flag==1)
                          {  t2=t2->down;
                          while(t2!=NULL)
                          {     cout<<"\n\t\t\t\t\t\tNAME OF SUBSECTION::  "<<t2->name;
                          t2=t2->next;
                          }
                          }
                          t1=t1->next;
                    }
                  }
                    temp=temp->next;
               }

              }
          }



}
int main()
{    Gll g;   int x;
     while(1)
     {    cout<<"\n\n enter your choice:";
          cout<<"\n 1.insert book:";
          cout<<"\n 2.insert chapter:";
          cout<<"\n 3.insert section:";
          cout<<"\n 4.insert subsection:";
          cout<<"\n 5.display book:";
          cout<<"\n 6.exit:";
          cin>>x;
         switch(x)
          {   case 1:        g.insertb();
                             break;
              case 2:        g.insertc();
                             break;
              case 3:        g.inserts();
                             break;
              case 4:        g.insertss();
                             break;
              case 5:        g.displayb();
                             break;
              case 6:  exit(0);
          }
     }
     return 0;
}
```

Output:


Activities
Terminal
May 15 15:59
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ g++ DSA1.cpp -o a
ads@ads-Veriton-M200-H81:~$ ./a
enter your choice:
1.insert book:
2.insert chapter:
3.insert section:
4.insert subsection:
5.display book:
6.exit:1
enter the name::C++_CODER
enter your choice:
1.insert book:
2.insert chapter:
3.insert section:
4.insert subsection:
5.display book:
6.exit:2
how many chapters you want to insert::2
enter the name::Introduction_to_c++
enter the name::Data_types_and_variables
enter your choice:
1.insert book:
2.insert chapter:
3.insert section:
4.insert subsection:
5.display book:
6.exit:3


Activities
Terminal
May 15 15:59
ads@ads-Veriton-M200-H81: ~
how many chapters you want to insert::2
enter the name::Introduction_to_c++
enter the name::Data_types_and_variables
enter your choice:
1.insert book:
2.insert chapter:
3.insert section:
4.insert subsection:
5.display book:
6.exit:3
Enter the name of chapter on which you want to enter the section::Introduction_to_c++
enter your choice:
1.insert book:
2.insert chapter:
3.insert section:
4.insert subsection:
5.display book:
6.exit:5
NAME OF BOOK: C++_CODER
NAME OF CHAPTER:: Introduction_to_c++
NAME OF CHAPTER:: Data_types_and_variables
enter your choice:
1.insert book:
2.insert chapter:
3.insert section:
4.insert subsection:
5.display book:
6.exit:6

Experiment No- 2

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 nodeii.Find number of nodes in longest path iii.Minimum data value found inthe tree iv.Change a tree so that the roles of the left and right pointers are swapped at every node v.Search a value.

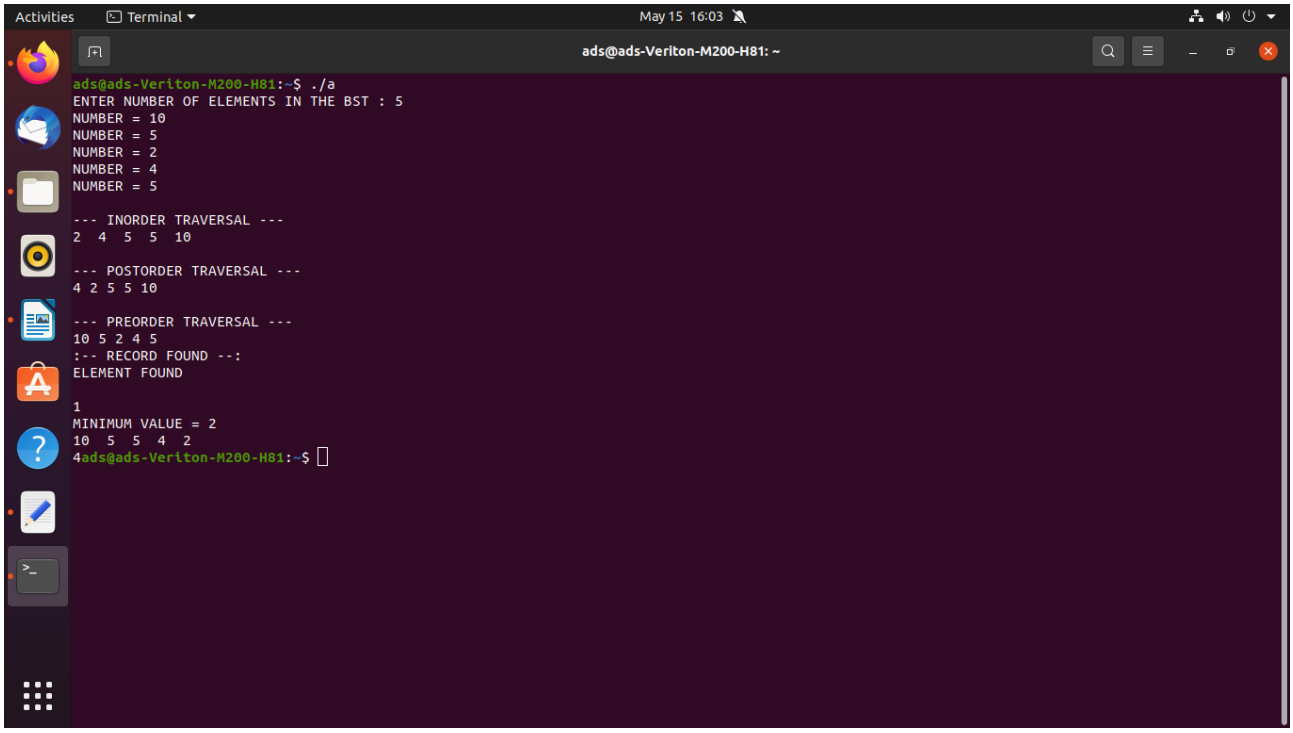
```
#include<iostream>
#include<math.h>
using namespace std;

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:


Activities
Terminal
May 15 16:03
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ ./a
ENTER NUMBER OF ELEMENTS IN THE BST : 5
NUMBER = 10
NUMBER = 5
NUMBER = 2
NUMBER = 4
NUMBER = 5
--- INORDER TRAVERSAL ---
2 4 5 5 10
--- POSTORDER TRAVERSAL ---
4 2 5 5 10
--- PREORDER TRAVERSAL ---
10 5 2 4 5
:-- RECORD FOUND --:
ELEMENT FOUND
1
MINIMUM VALUE = 2
10 5 5 4 2
4ads@ads-Veriton-M200-H81:~$

Experiment No.-3
For given expression eg. a-b*c-d/e+f construct inorder sequence and traverse it using postorder traversal(non recursive).

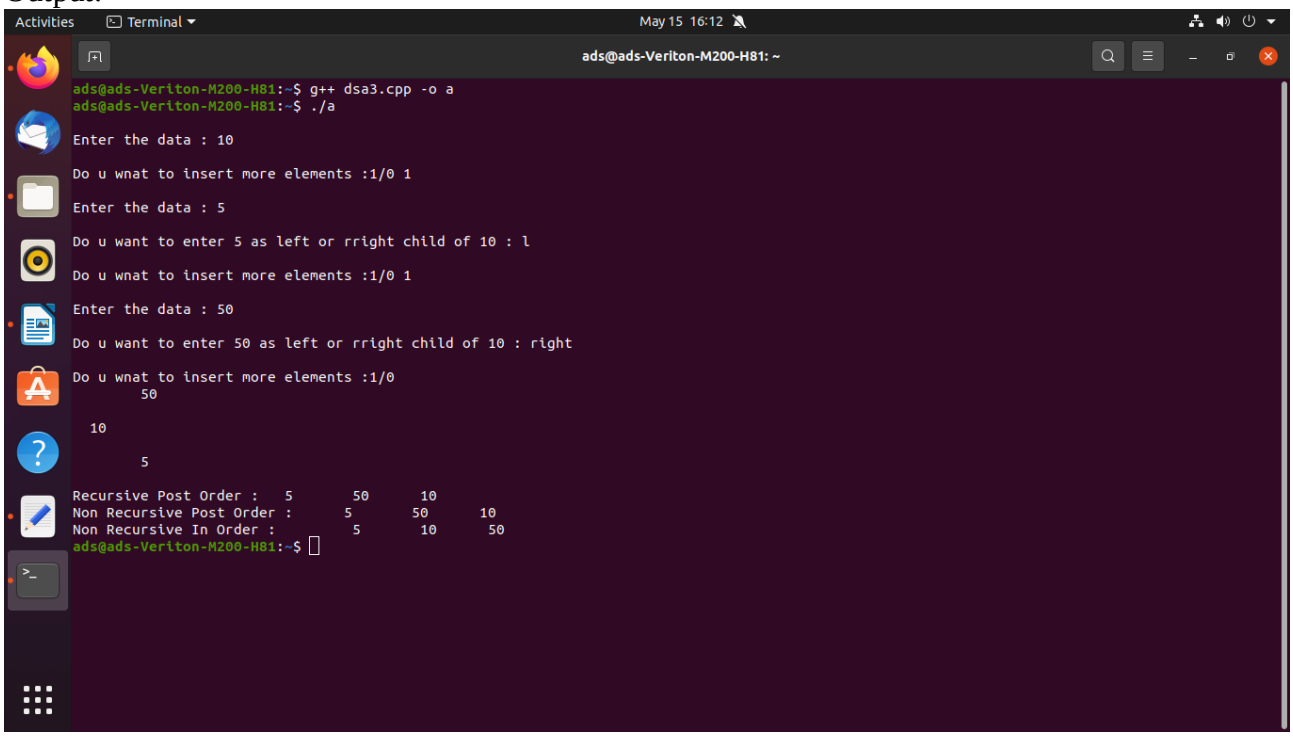
```
#include<iostream>
#include<stack>
using namespace std;

class Btree{
 typedef struct node{
  int data;
  struct node * right ,*left;
 }node;

public :
 node * root,*temp;
 Btree(){
  root=new node;
  root=NULL;
 }
 void create(){
  temp=new node;

  cout<<"\nEnter the data : ";
  cin>>temp->data;
  temp->left=temp->right=NULL;
  if(root==NULL){
   root=temp;
  }
  else{
   insert(root,temp);
  }
 }

 void insert(node * root, node * temp){
  char ch;
  cout<<"\nDo u want to enter "<<temp->data<<" as left or rright child of "<<root->data<<" : ";
  cin>>ch;
  if(ch=='l'){
   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 postOrder_recursive(node * root){
 if(root!=NULL){
  postOrder_recursive(root->left);
  postOrder_recursive(root->right);
  cout<<"\t "<<root->data;
 }
}
void postOrder_nonRecursive(node* root){
 if(!root)
 {
  cout<<"\nEmpty";
  return;
 }
 stack<node *> s;
 stack<node*> p;
 s.push(root);
 while(!s.empty()){
  node * cur=s.top();
  p.push(cur);
  s.pop();
  if(cur->left)
   s.push(cur->left);
  if(cur->right)
   s.push(cur->right);
 }
 while(!p.empty()){
  cout<<"\t"<<p.top()->data;
  p.pop();  }
}
//Extra Part : Inorder
 void InOrder_nonRecursive(node* root){
  if(!root)
  {
   cout<<"\nEmpty";
   return;
  }
  stack<node *> s;
  stack<node*> p;

  while(true){
   while(root!=NULL){
    //cout<<"\t "<<root->data;
    s.push(root);
    root=root->left;
   }
   if(s.empty())
    return;
   root=s.top();
   s.pop();

   cout<<"\t "<<root->data;
```

```
    root=root->right;
     }
  }
 //Extra Part : PreOrder
 void PreOrder_nonRecursive(node* root){
    if(!root)
    {
     cout<<"\nEmpty";
     return;
    }
    stack<node *> s;
    stack<node*> p;

    while(true){
     while(root!=NULL){
      cout<<"\t "<<root->data;
      s.push(root);
      root=root->left;
     }
     if(s.empty())
      return;
     root=s.top();
     s.pop();

     //cout<<"\t "<<root->data;
     root=root->right;
      }
   }
 void display(node* root, int space){

  if(root==NULL)
   return ;
  space +=3;
  display(root->right,space);
  cout<<"\n";
  for(int i=3; i<=space ; i++){
   cout<<"  ";}
   cout<<root->data<<"\n";
  display(root->left,space);
 }

};
int main(){
Btree b;
int ch;
do{
 b.create();
 cout<<"\nDo u wnat to insert more elements :1/0 ";
 cin>>ch;

}while(ch!=0);
b.display(b.root,0);
```

```
cout<<"\nRecursive Post Order : ";
b.postOrder_recursive(b.root);
cout<<"\nNon Recursive Post Order : ";
b.postOrder_nonRecursive(b.root);
cout<<"\nNon Recursive In Order : ";
b.InOrder_nonRecursive(b.root);
cout<<"\nNon Recursive Pre Order : ";
b.PreOrder_nonRecursive(b.root);
 return 0;
}
```

Output:



Experiment No.-4

Convert given binary tree into threaded binary tree. Analyze time and space complexity of the algorithm.

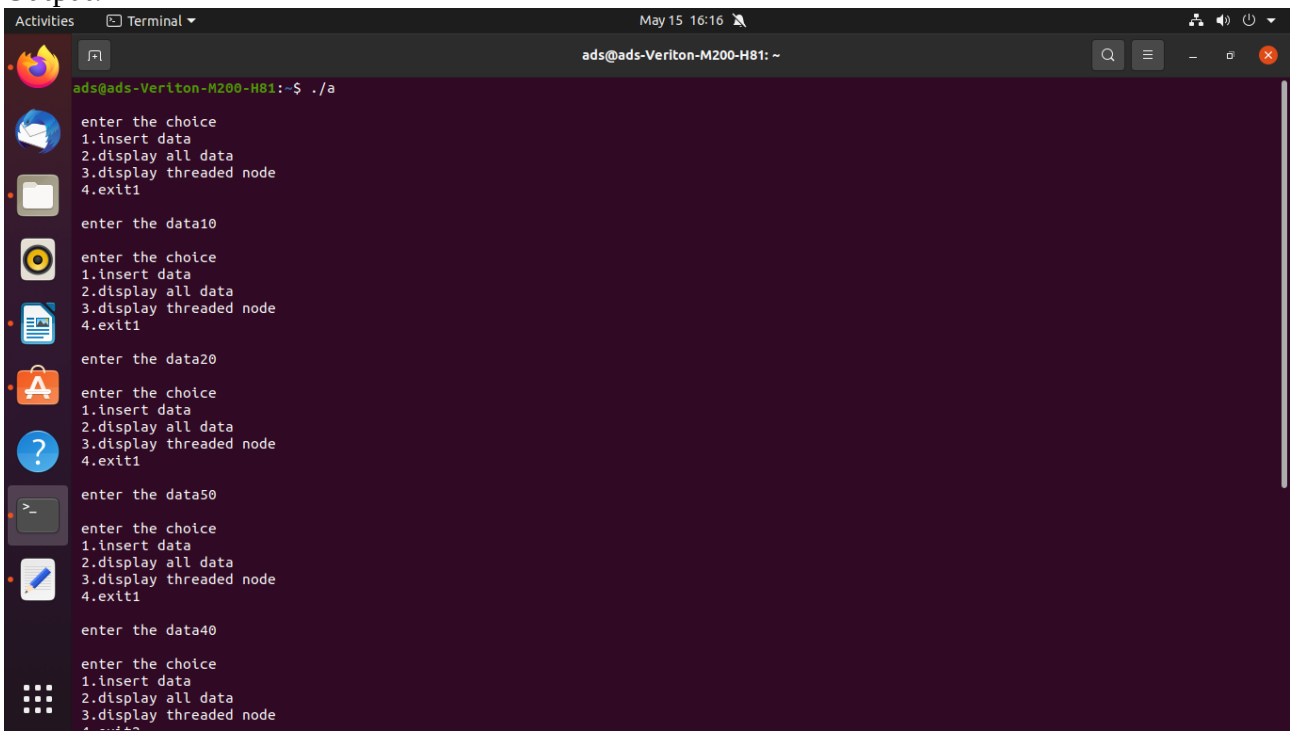
```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct node
{
    int data;
    node *left,*right;
    int lbit,rbit;
};
class tbt
{
   node *temp=NULL,*t1=NULL,*s=NULL,*head=NULL,*t=NULL;
   public:

   node *create();
   void insert();
   node *insuc(node*);
   node *inpre(node*);
   void dis();
   void display(node*);
   void thr();
   void thread(node*);
};
node *tbt::create()
{
   node *p=new(struct node);
   p->left=NULL;
   p->right=NULL;
   p->lbit=0;
   p->rbit=0;
   cout<<"\n enter the data";
   cin>>p->data;
   return p;
}
void tbt::insert()
{
    temp=create();
    if(head==NULL)
    {  node *p=new(struct node);
       head=p;
       head->left=temp;
       head->right=head;
       head->lbit=1;
       head->rbit=0;
       temp->left=head;
       temp->right=head;
       temp->lbit=0;
```

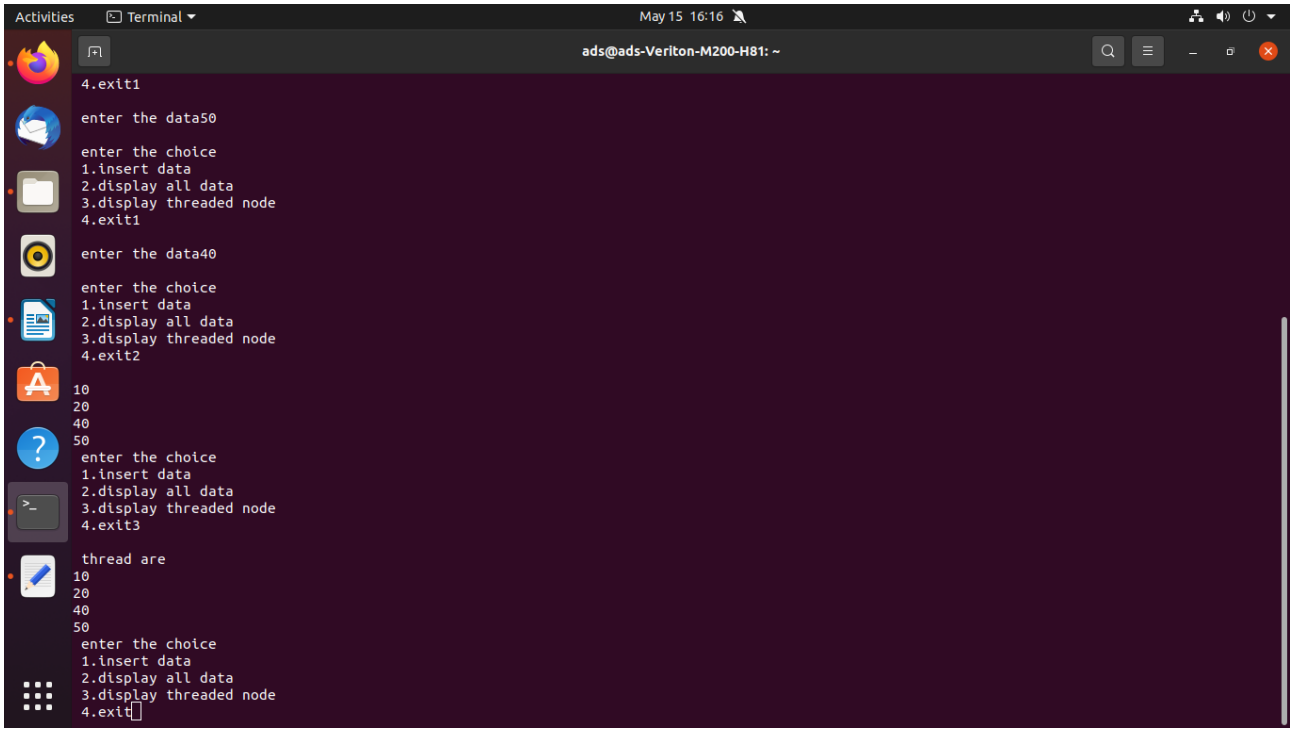
```
      temp->rbit=0;
    }
    else
    {     t1=head;
          t1=t1->left;

          while(t1!=NULL)
          {   s=t1;
              if(((temp->data)>(t1->data))&&t1->rbit==1)
              {   t1=t1->right;   }
              else if(((temp->data)<(t1->data))&&t1->lbit==1)
              {   t1=t1->left;     }
              else
              { break;       }
          }
          if(temp->data>s->data)
          {
            s->right=temp;
            s->rbit=1;
            temp->left=inpre(head->left);
            temp->right=insuc(head->left);
          }
          else
          {
            s->left=temp;
            s->lbit=1;
            temp->left=inpre(head->left);
            temp->right=insuc(head->left);
          }
     }




}
node *tbt::inpre(node *m)
{
    if(m->lbit==1)
    {
      inpre(m->left);
    }
    if(m->data==temp->data&&t==NULL)
    {  return head;       }
    if(m->data==temp->data)
    {   return t;         }
    t=m;
    if(m->rbit==1)
    { inpre(m->right);
    }

}
node *tbt::insuc(node *m)
```

```
{
   if(m->lbit==1)
   {  t=m;
     insuc(m->left);
   }

   if(m->data==temp->data&&t==NULL)
   {  return head;       }
   if(m->data==temp->data)
   {   return t;        }

   if(m->rbit==1)
   { insuc(m->right);
   }
}
void tbt::dis()
{   display(head->left);
}
void tbt::display(node *m)
{

      if(m->lbit==1)
      {   display(m->left);              }
      cout<<"\n"<<m->data;
      if(m->rbit==1)
      {    display(m->right);            }



}
void tbt::thr()
{  cout<<"\n thread are";
  thread(head->left);
}
void tbt::thread(node *m)
{

      if(m->lbit==1)
      {   thread(m->left);              }
      if(m->lbit==0||m->rbit==0)
      {
      cout<<"\n"<<m->data;
      }
      if(m->rbit==1)
      {    thread(m->right);            }



}
int main()
{  tbt t;  int ch;
  while(1)
```

```
  {

  cout<<"\n enter the choice";
  cout<<"\n 1.insert data";
  cout<<"\n 2.display all data";
  cout<<"\n 3.display threaded node";
  cout<<"\n 4.exit";
  cin>>ch;
  switch(ch)
  {
    case 1:
            t.insert();
            break;
    case 2:
            t.dis();
            break;
    case 3:
            t.thr();
            break;
    case 4:  exit(0);

    default:
          cout<<"\n invalid entry";
   }
   }
  return 0;
}
```

Output:


Activities
Terminal
May 15 16:16
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ ./a
enter the choice
1.insert data
2.display all data
3.display threaded node
4.exit1
enter the data10
enter the choice
1.insert data
2.display all data
3.display threaded node
4.exit1
enter the data20
enter the choice
1.insert data
2.display all data
3.display threaded node
4.exit1
enter the data50
enter the choice
1.insert data
2.display all data
3.display threaded node
4.exit1
enter the data40
enter the choice
1.insert data
2.display all data
3.display threaded node


Activities
Terminal
May 15 16:16
ads@ads-Veriton-M200-H81: ~
4.exit1
enter the data50
enter the choice
1.insert data
2.display all data
3.display threaded node
4.exit1
enter the data40
enter the choice
1.insert data
2.display all data
3.display threaded node
4.exit2
10
20
40
50
enter the choice
1.insert data
2.display all data
3.display threaded node
4.exit3
thread are
10
20
40
50
enter the choice
1.insert data
2.display all data
3.display threaded node
4.exit

Experiment No.5

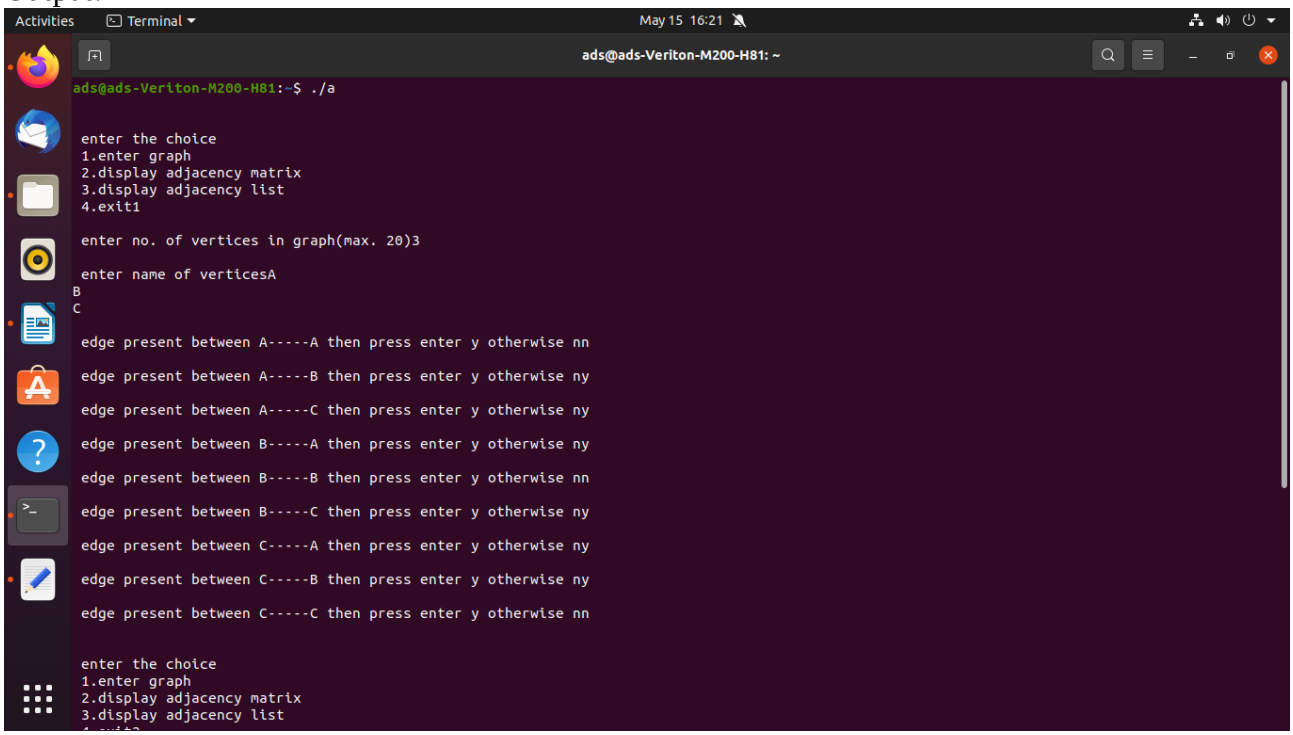
Write a function to get the number of vertices in an undirected graph and its edges. You may assume that no edge is input twice. i. Use adjacency list representation of the graph and find runtime of the function ii. Use adjacency matrix representation of the graph and find runtime of the function.

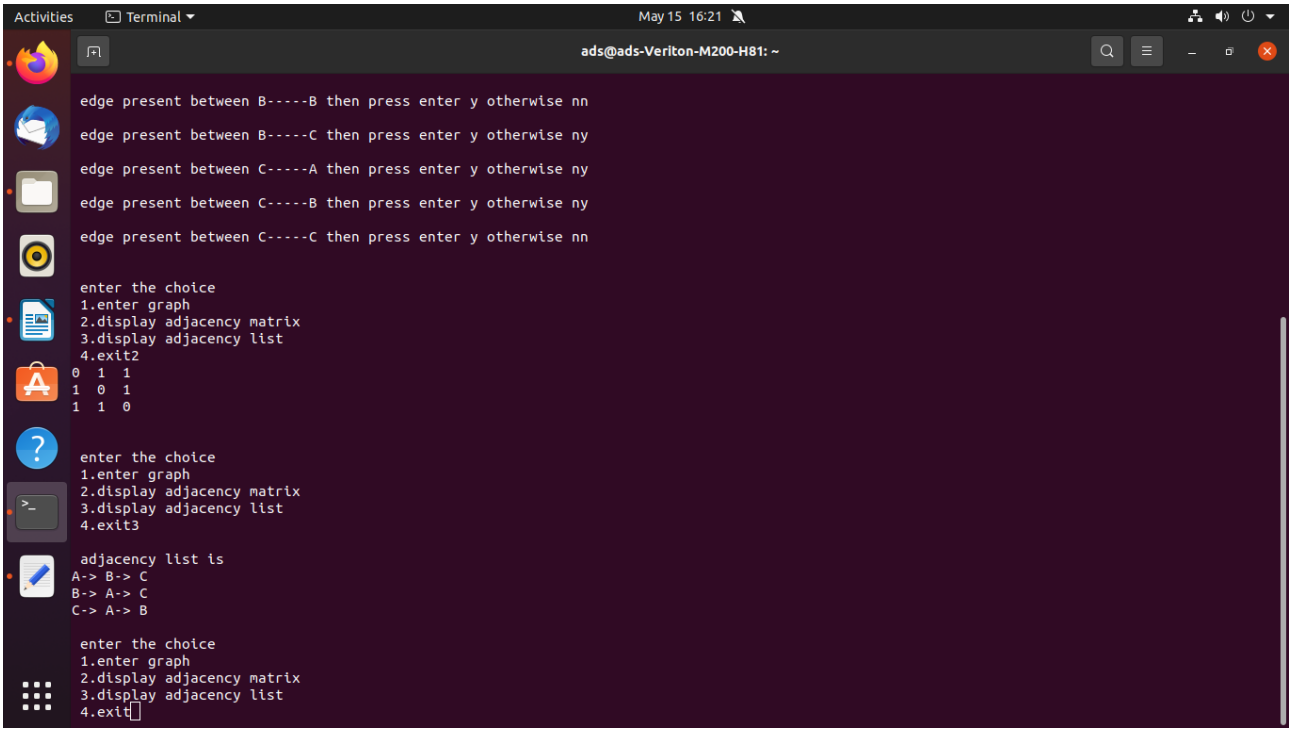
```
#include<iostream>
#include<stdlib.h>
using namespace std;
struct node
{   char vertex;
   node *next;
};
class adjmatlist
{    int m[10][10],n,i,j; char ch;  char v[20];   node *head[20];  node *temp=NULL;

     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 vertices in graph(max. 20)";
  cin>>n;
  cout<<"\n enter name of vertices";
  for(i=0;i<n;i++)
    cin>>v[i];
  for(i=0;i<n;i++)
  {
     for(j=0;j<n;j++)
     {  cout<<"\n edge present between "<<v[i]<<"-----"<<v[j]<<" then press enter y otherwise n";
        cin>>ch;
        if(ch=='y')
        { m[i][j]=1;  }
        else if(ch=='n')
        {  m[i][j]=0;  }
        else
        { cout<<"\n unknown entry";  }
     }
  }
     adjlist();

}
void adjmatlist::adjlist()
{
```

```
      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]==1)
               {
                    node *p=new(struct node);
                    p->vertex=v[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()
{
    for(i=0;i<n;i++)
    {  for(j=0;j<n;j++)
       cout<<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;  }

                    }




      }

}
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";
   cout<<"\n 3.display adjacency list";
   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:


Activities
Terminal
May 15 16:21
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ ./a
enter the choice
1.enter graph
2.display adjacency matrix
3.display adjacency list
4.exit1
enter no. of vertices in graph(max. 20)3
enter name of verticesA
B
C
edge present between A-----A then press enter y otherwise nn
edge present between A-----B then press enter y otherwise ny
edge present between A-----C then press enter y otherwise ny
edge present between B-----A then press enter y otherwise ny
edge present between B-----B then press enter y otherwise nn
edge present between B-----C then press enter y otherwise ny
edge present between C-----A then press enter y otherwise ny
edge present between C-----B then press enter y otherwise ny
edge present between C-----C then press enter y otherwise nn
enter the choice
1.enter graph
2.display adjacency matrix
3.display adjacency list


Activities
Terminal
May 15 16:21
ads@ads-Veriton-M200-H81: ~
edge present between B-----B then press enter y otherwise nn
edge present between B-----C then press enter y otherwise ny
edge present between C-----A then press enter y otherwise ny
edge present between C-----B then press enter y otherwise ny
edge present between C-----C then press enter y otherwise nn
enter the choice
1.enter graph
2.display adjacency matrix
3.display adjacency list
4.exit2
0 1 1
1 0 1
1 1 0
enter the choice
1.enter graph
2.display adjacency matrix
3.display adjacency list
4.exit3
adjacency list is
A-> B-> C
B-> A-> C
C-> A-> B
enter the choice
1.enter graph
2.display adjacency matrix
3.display adjacency list
4.exit

Experiment No.-6

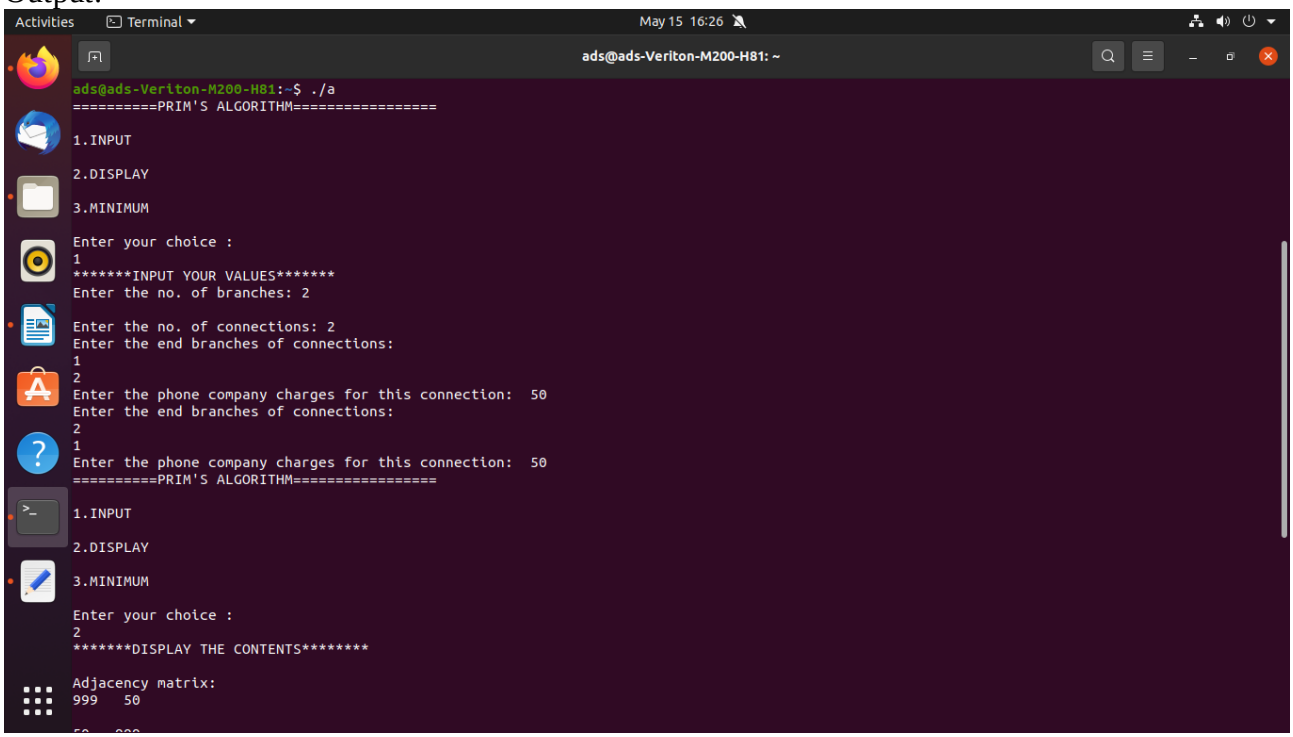
You have a business with several offices; you want to lease phone lines to connect them up with each other; and the phone company charges different amounts of money to connect different pairs of cities. You want a set of lines that connects all your offices with minimum total cost. Solve the problem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

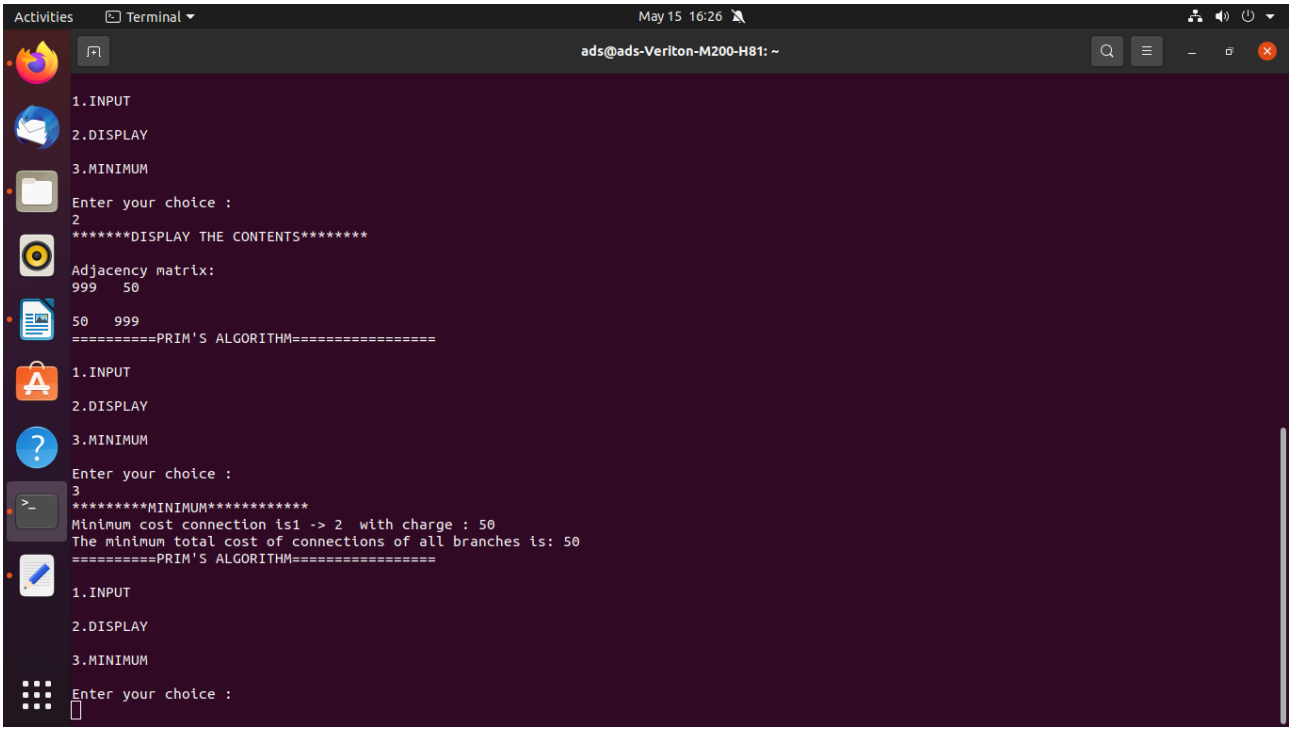
```
#include<iostream>
using namespace std;

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:


Activities
Terminal
May 15 16:26
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ ./a
==========PRIM'S ALGORITHM==========
1.INPUT
2.DISPLAY
3.MINIMUM
Enter your choice :
1
*******INPUT YOUR VALUES*******
Enter the no. of branches: 2
Enter the no. of connections: 2
Enter the end branches of connections:
1
2
Enter the phone company charges for this connection: 50
Enter the end branches of connections:
2
1
Enter the phone company charges for this connection: 50
==========PRIM'S ALGORITHM==========
1.INPUT
2.DISPLAY
3.MINIMUM
Enter your choice :
2
*******DISPLAY THE CONTENTS********
Adjacency matrix:
999 50


Activities
Terminal
May 15 16:26
ads@ads-Veriton-M200-H81: ~
1.INPUT
2.DISPLAY
3.MINIMUM
Enter your choice :
2
*******DISPLAY THE CONTENTS********
Adjacency matrix:
999 50
50 999
==========PRIM'S ALGORITHM==========
1.INPUT
2.DISPLAY
3.MINIMUM
Enter your choice :
3
*********MINIMUM***********
Minimum cost connection is1 -> 2 with charge : 50
The minimum total cost of connections of all branches is: 50
==========PRIM'S ALGORITHM==========
1.INPUT
2.DISPLAY
3.MINIMUM
Enter your choice :

Experiment No.-7

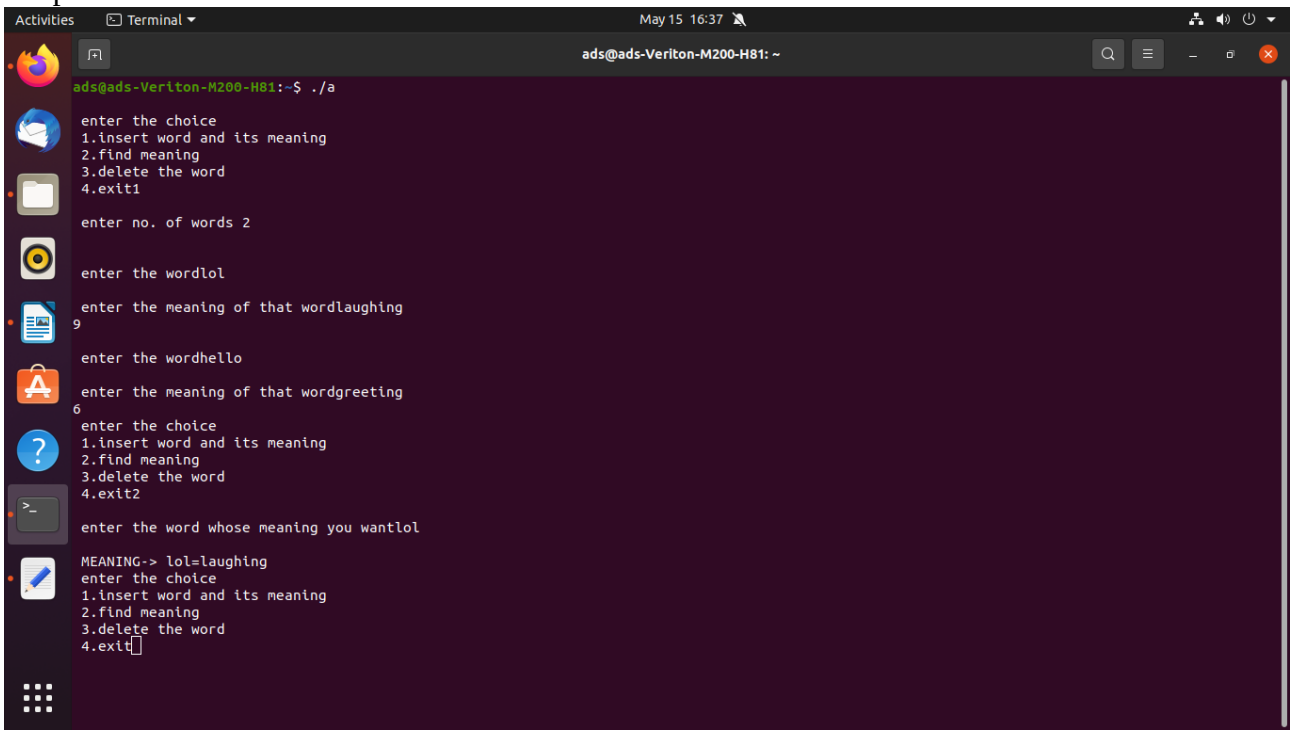
Implement all the functions of a dictionary (ADT) using hashing. Data: Set of (key, value) pairs, Keys are mapped to values, Keys must be comparable, Keys must be unique Standard Operations: Insert (key, value), Find (key), Delete (key).

```
#include<iostream>
#include<string.h>
#include<stdlib.h>
using namespace std;
struct data
{  char name[30];
   char name1[30];
};
class hash
{  int n,sum,x,c,i,j;   char na[30],na1[30];
   data d[10];
   public:
   hash()
   {  for(i=0;i<10;i++)
      {  strcpy(d[i].name,"\0");   }
   }
   void insert();
   void search();
   void delet();
   void display();
};
void hash::insert()
{
  cout<<"\n enter no. of words";
  cin>>n;
  for(j=0;j<n;j++)
  {   cout<<"\n\n enter the word";
      cin>>na;
      cout<<"\n enter the meaning of that word";
      cin>>na1;
      sum=0;
      for(i=0;i<strlen(na);i++)
      {   sum=sum+na[i];
      }
      x=(sum/strlen(na))%10;
      cout<<x;
      c=x;
          while(1)
          {

              if(!strcmp(d[x].name,"\0"))
              {   strcpy(d[x].name,na);
                  strcpy(d[x].name1,na1);
                  break;
              }
               x=(x+1)%10;
```

```
            if(c==x)
            {  cout<<"\n hash table is full";
               break;
            }
          }

   }
}
void hash::search()
{  cout<<"\n enter the word whose meaning you want";
   cin>>na;

    sum=0;
      for(i=0;i<strlen(na);i++)
      {   sum=sum+(int)na[i];
      }
      x=(sum/strlen(na))%10;
      c=x;
   while(1)
         {

            if(!strcmp(d[x].name,na))
            {   cout<<"\n MEANING-> "<<d[x].name<<"="<<d[x].name1;
                break;
            }
             x=(x+1)%10;
            if(c==x)
            {  cout<<"\n word not found";
               break;
            }
         }
}
void hash::delet()
{  cout<<"\n enter the word which is to be deleted";
   cin>>na;

    sum=0;
      for(i=0;i<strlen(na);i++)
      {   sum=sum+(int)na[i];
      }
      x=(sum/strlen(na))%10;
      c=x;
   while(1)
         {

            if(!strcmp(d[x].name,na))
            {   cout<<"\n"<<d[x].name<<" word deleted";
               strcpy(d[x].name,"\0");  strcpy(d[x].name1,"\0");
               break;
            }
             x=(x+1)%10;
            if(c==x)
```

```
            {  cout<<"\n word not found";
               break;
            }
          }
}
void hash::display()
{
       for(int i=0;i<10;i++){
       cout<<endl<<d[i].name<<" "<<d[i].name1;
       }


}
int main()
{
  hash h;  int n;
  while(1)
  {
     cout<<"\n enter the choice";
     cout<<"\n 1.insert word and its meaning";
     cout<<"\n 2.find meaning";
     cout<<"\n 3.delete the word";
     cout<<"\n 4.exit";
     cin>>n;
     switch(n)
     {
       case 1:   h.insert();
                 break;
       case 2:   h.search();
                 break;
       case 3:   h.delet();
                 break;
       case 4: exit(0);
       default: cout<<"\n unknown choice";
     }
  }
     return 0;
}
```

Output:


Activities
Terminal
May 15 16:37
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ ./a
enter the choice
1.insert word and its meaning
2.find meaning
3.delete the word
4.exit1
enter no. of words 2
enter the wordlol
enter the meaning of that wordlaughing
9
enter the wordhello
enter the meaning of that wordgreeting
6
enter the choice
1.insert word and its meaning
2.find meaning
3.delete the word
4.exit2
enter the word whose meaning you wantlol
MEANING-> lol=laughing
enter the choice
1.insert word and its meaning
2.find meaning
3.delete the word
4.exit

Experiment No.-8

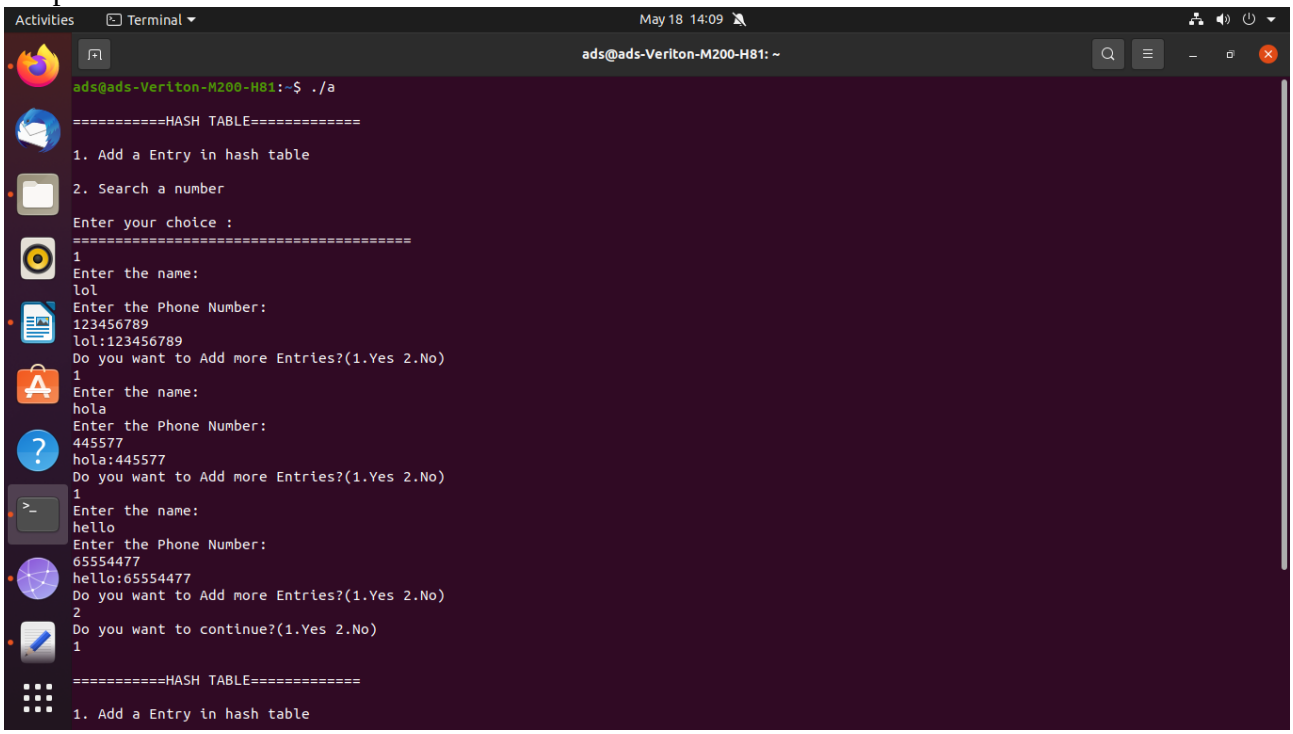
Consider telephone book database of N clients. Make use of a hash table implementation to quickly lookup client's phone number.

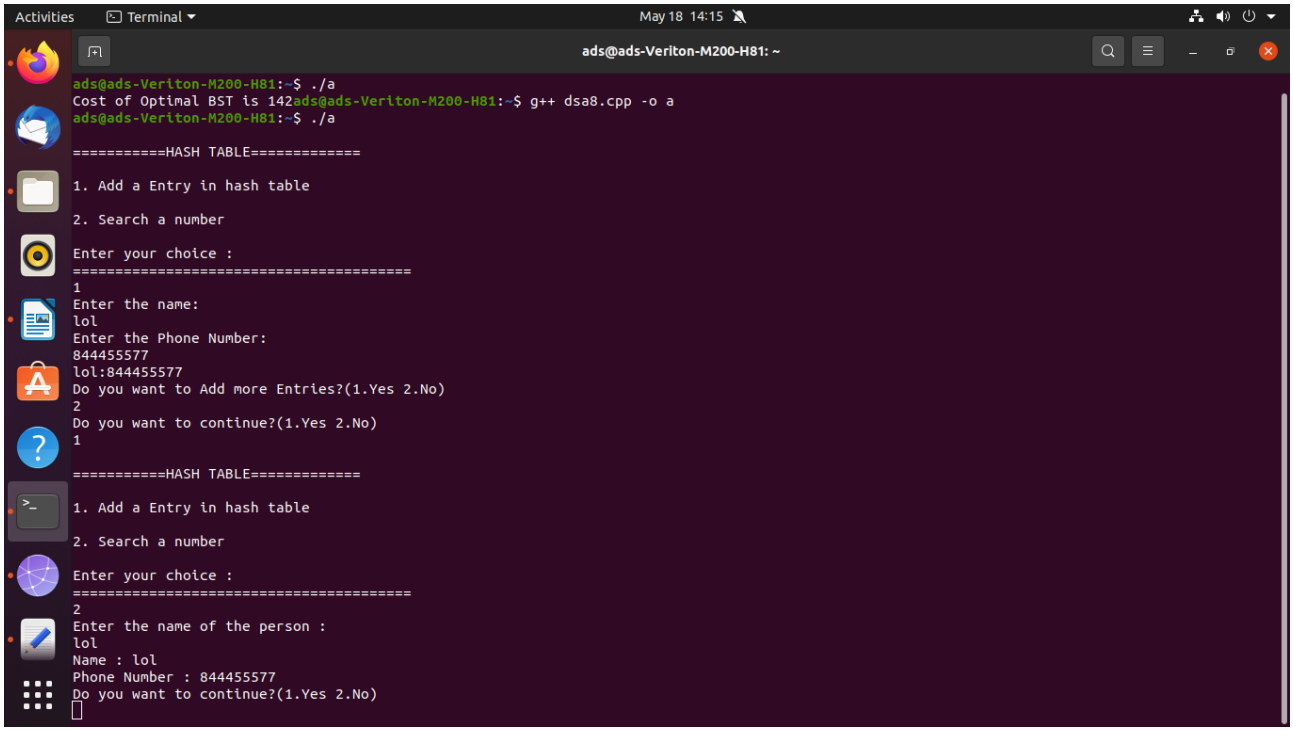
```
#include<iostream>
using namespace std;

#define KEY(x) x%10

        struct node
        {
        string name,phnum;
        int key;
        node *next;
        }*hash[10];


        class Database
        {
        public:

        int sum(string name)
                {
                int sum =0;
                for(int i=0;name[i]!='\0';i++)
                   sum = sum + name[i];
                return sum;
                }


        node* create()
        {
        node *temp = new (struct node);
        cout<<"Enter the name:"<<endl;
        cin.ignore();
        getline(cin,temp->name);

        cout<<"Enter the Phone Number:"<<endl;
        getline(cin,temp->phnum);
        temp->next='\0';
        int z=sum(temp->name);
        temp->key = KEY(z);
        return temp;
        }


        void position(node *p,int key)
        {
                if(!hash[key])
                {
                hash[key] = p;
```

```
        }
        else
        {
        node *q;
        q=hash[key];
        while(q->next!='\0')
        q=q->next;
        q->next = p;
        }
}


void add()
{
node *p;
p = create();
position(p,p->key);
        while(p!=NULL)
        {
        cout<<p->name<<":"<<p->phnum<<" "<<"\n";
        p=p->next;
        }

}


node* search(int key,string name)
{
node *p;
p=hash[key];
while((p!='\0')&&(name!=p->name))
   p=p->next;
return p;
}

};

int main()
{
Database obj;
node n1;
string a;
int key,x;
int ch;
do
{
cout<<"\n===========HASH TABLE============"<<endl;
cout<<"\n1. Add a Entry in hash table"<<endl;
cout<<"\n2. Search a number"<<endl;
cout<<"\nEnter your choice :"<<endl;
cout<<"========================================"<<endl;
cin>>ch;
```

```
        switch(ch)
        {
        case 1:
        {
        do
        {
         obj.add();
         cout<<"Do you want to Add more Entries?(1.Yes 2.No)"<<endl;
         cin>>x;
        }while(x==1);

         break;
        }
        case 2:
        {
         cout<<"Enter the name of the person : "<<endl;
         cin.ignore();
         getline(cin,a);
         int z=obj.sum(a);
         key = KEY(z);
         node *p;
         p=obj.search(key,a);
         if(!p)
                 cout<<"No such Entry in DataBase"<<endl;
        else
        {
        cout<<"Name : "<<p->name<<endl;
               cout<<"Phone Number : "<<p->phnum<<endl;
        }
        break;
        }

        default:cout<<"Invalid Entry"<<endl;
        break;
        }
        cout<<"Do you want to continue?(1.Yes 2.No)"<<endl;
        cin>>x;
        }while(x==1);

        return 0;
}
```

Output:


Activities
Terminal
May 18 14:09
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ ./a
==========HASH TABLE=============
1. Add a Entry in hash table
2. Search a number
Enter your choice :
=======================================
1
Enter the name:
lol
Enter the Phone Number:
123456789
lol:123456789
Do you want to Add more Entries?(1.Yes 2.No)
1
Enter the name:
hola
Enter the Phone Number:
445577
hola:445577
Do you want to Add more Entries?(1.Yes 2.No)
1
Enter the name:
hello
Enter the Phone Number:
65554477
hello:65554477
Do you want to Add more Entries?(1.Yes 2.No)
2
Do you want to continue?(1.Yes 2.No)
1
==========HASH TABLE=============
1. Add a Entry in hash table


Activities
Terminal
May 18 14:15
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ ./a
Cost of Optimal BST is 142ads@ads-Veriton-M200-H81:~$ g++ dsa8.cpp -o a
ads@ads-Veriton-M200-H81:~$ ./a
==========HASH TABLE=============
1. Add a Entry in hash table
2. Search a number
Enter your choice :
=======================================
1
Enter the name:
lol
Enter the Phone Number:
844455577
lol:844455577
Do you want to Add more Entries?(1.Yes 2.No)
2
Do you want to continue?(1.Yes 2.No)
1
==========HASH TABLE=============
1. Add a Entry in hash table
2. Search a number
Enter your choice :
=======================================
2
Enter the name of the person :
lol
Name : lol
Phone Number : 844455577
Do you want to continue?(1.Yes 2.No)

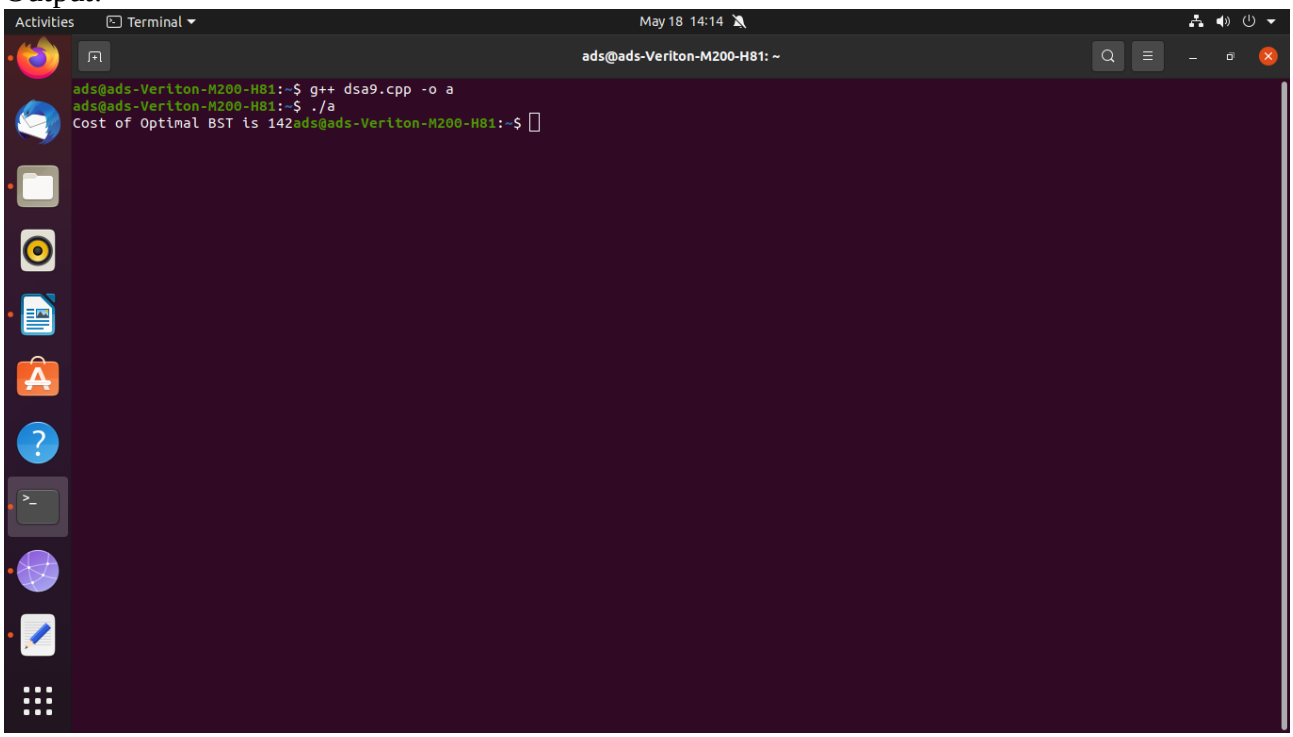
Experiment No: 9

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.

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

int sum(int frequency[], int i, int j)
{
    int sum = 0;
    for (int x = i; x <= j; x++)
        sum += frequency[x];
    return sum;
}

int optimalCost(int frequency[], int i, int j)
{
    if (j < i)
        return 0;
    if (j == i)
        return frequency[i];

    int frequencySum = sum(frequency, i, j);

    int min = INT_MAX;

    for (int r = i; r <= j; ++r)
    {
        int cost = optimalCost(frequency, i, r - 1) + optimalCost(frequency, r + 1, j);
        if (cost < min)
            min = cost;
    }

    return min + frequencySum;
}

int optimalSearchTree(int keys[], int frequency[], int n)
{
    return optimalCost(frequency, 0, n - 1);
}

int main()
{
    int keys[] = {10, 12, 20};
    int frequency[] = {34, 8, 50};

    int n = sizeof(keys) / sizeof(keys[0]);

    cout << "Cost of Optimal BST is " << optimalSearchTree(keys, frequency, n);
```

```
    return 0;
}
```

Output:



Experiment No:10

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)
```

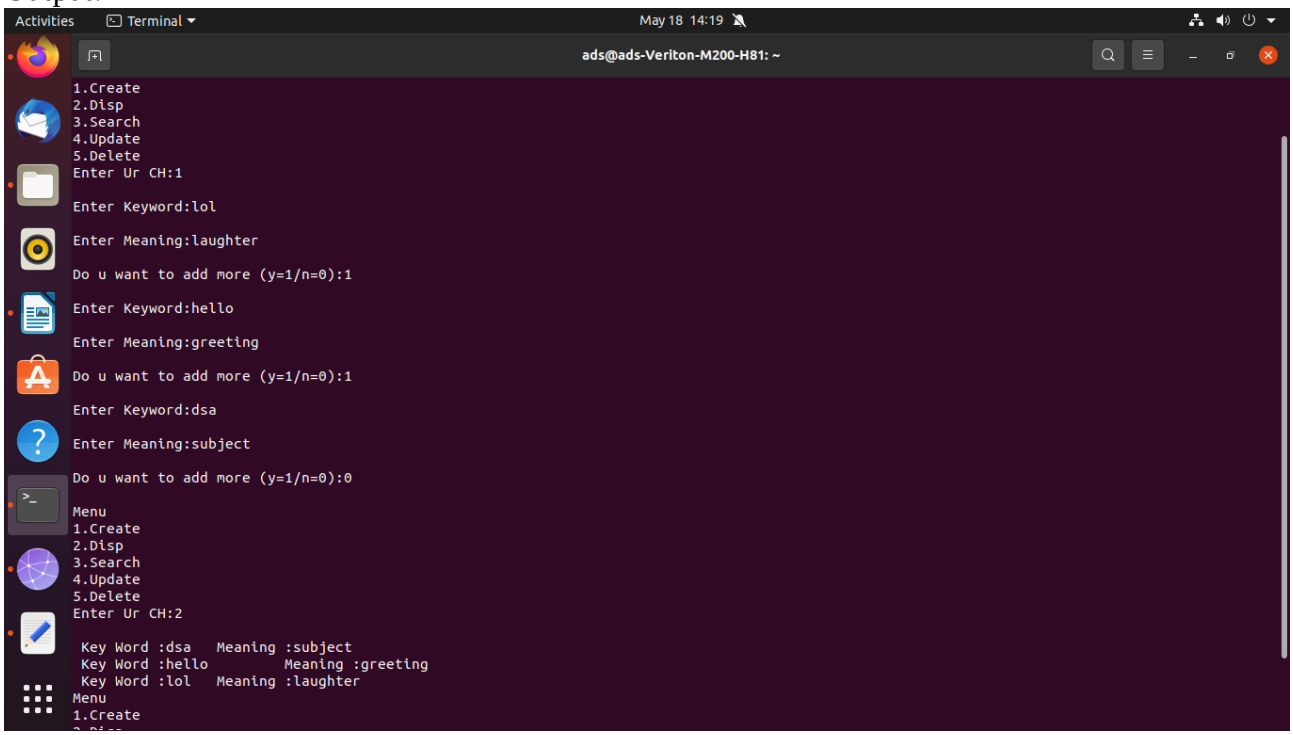
```
   root = root->left;
  if(strcmp (k, root->k) > 0)
   root = root->right;
 }

 return -1;
}
int dict :: update(node * root,char k[20])
{
 while(root != NULL)
 {
  if(strcmp (k,root->k) == 0)
  {
   cout<<"\nEnter New Meaning ofKeyword"<<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 No 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)
```

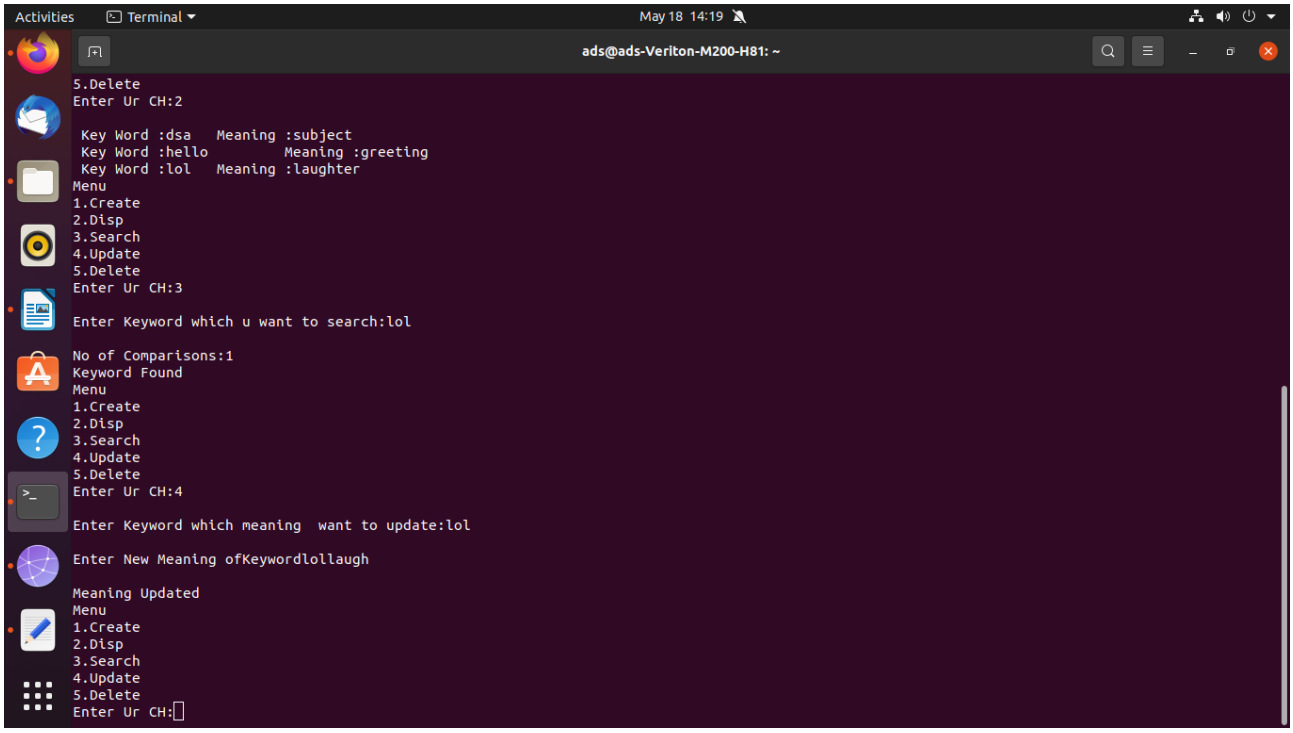
```
  {
  temp = root;
  root = root->left;
  delete temp;
  return root;
  }
  else if(root->left==NULL)
  {
  temp = root;
  root = root->right;
  delete temp;
  return root;
  }
  temp = min(root->right);
  strcpy(root->k,temp->k);
  root->right = del(root->right, temp->k);
  return root;

}

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 Ur CH:";
  cin>>ch;

  switch(ch)
  {
case 1: d.create();
  break;
case 2: if(d.root == NULL)
  {
  cout<<"\nNo any 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 which meaning  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:


Activities
Terminal
May 18 14:19
ads@ads-Veriton-M200-H81: ~
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter Ur CH:1
Enter Keyword:lol
Enter Meaning:laughter
Do u want to add more (y=1/n=0):1
Enter Keyword:hello
Enter Meaning:greeting
Do u want to add more (y=1/n=0):1
Enter Keyword:dsa
Enter Meaning:subject
Do u want to add more (y=1/n=0):0
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter Ur CH:2
Key Word :dsa Meaning :subject
Key Word :hello Meaning :greeting
Key Word :lol Meaning :laughter
Menu
1.Create


Activities
Terminal
May 18 14:19
ads@ads-Veriton-M200-H81: ~
5.Delete
Enter Ur CH:2
Key Word :dsa Meaning :subject
Key Word :hello Meaning :greeting
Key Word :lol Meaning :laughter
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter Ur CH:3
Enter Keyword which u want to search:lol
No of Comparisons:1
Keyword Found
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter Ur CH:4
Enter Keyword which meaning want to update:lol
Enter New Meaning ofKeywordlollaugh
Meaning Updated
Menu
1.Create
2.Disp
3.Search
4.Update
5.Delete
Enter Ur CH:

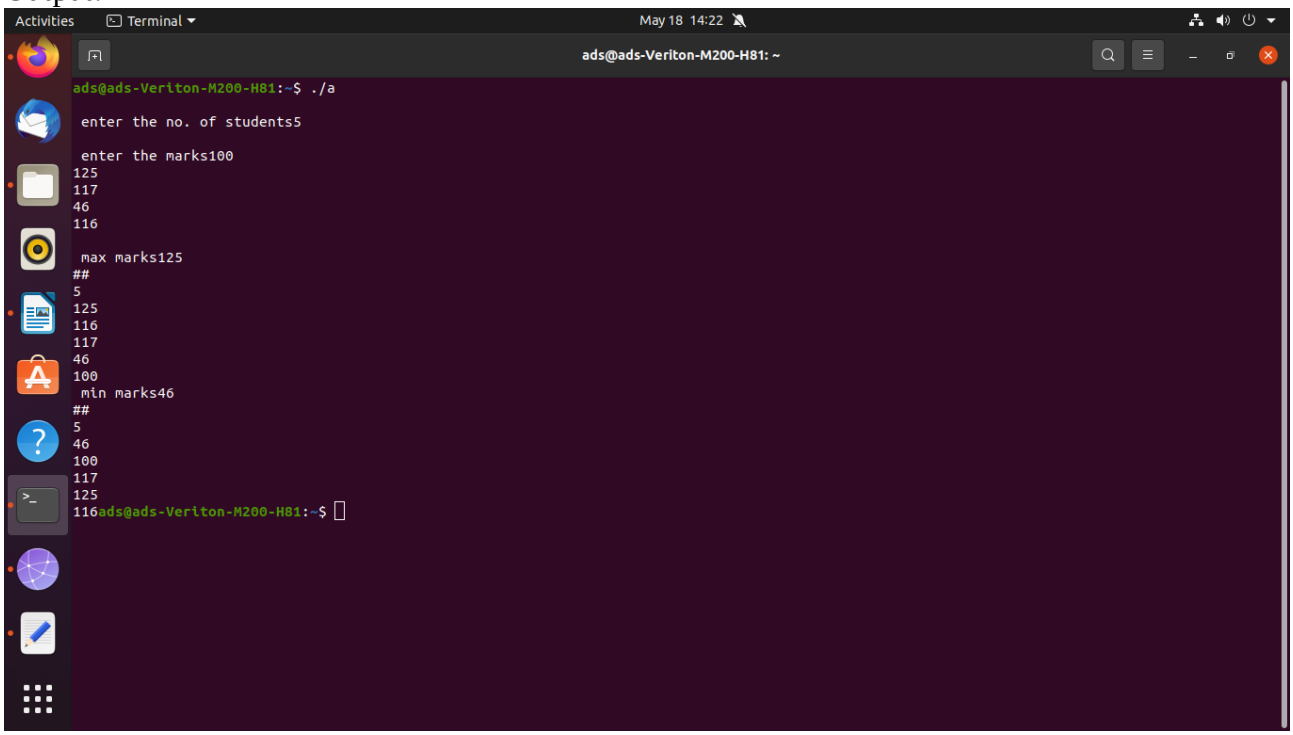
Experiment No. 11

Read the marks obtained by students of second year in an online examination of particular subject. Find out maximum and minimum marks obtained in a that subject. Use heap data structure. Analyse the algorithm

```
#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:


Activities
Terminal
May 18 14:22
ads@ads-Veriton-M200-H81: ~
ads@ads-Veriton-M200-H81:~$ ./a
enter the no. of students5
enter the marks100
125
117
46
116
max marks125
##
5
125
116
117
46
100
min marks46
##
5
46
100
117
125
116ads@ads-Veriton-M200-H81:~$

Experiment No: 12

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 the Name : ";
  cin>>name;
  cout<<"\n\tEnter the Division:";
  cin>>div;
  cout<<"\n\tEnter the 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()
{
```

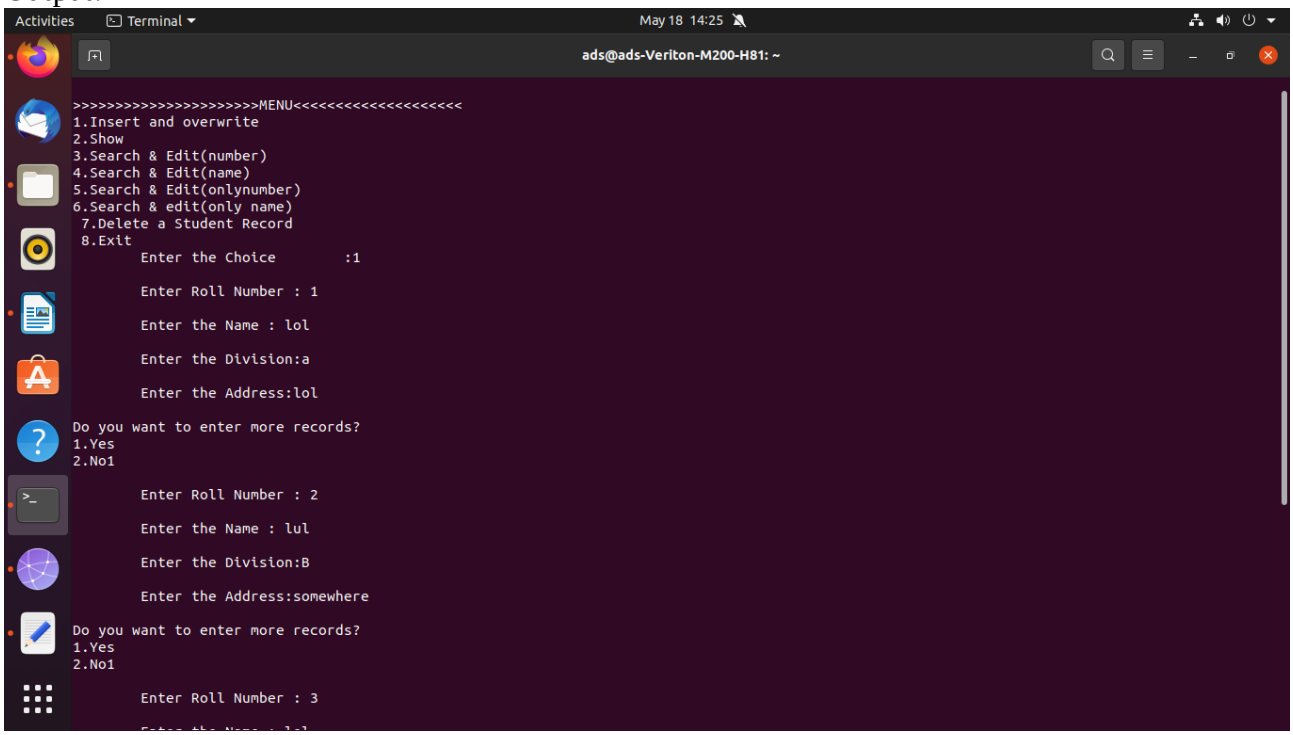
```
 int i,n,ch,ch1,rec,start,count,add,n1,add2,start2,n2,y,a,b,on,oname,add3,start3,n3,y1,add4,start4,n4;
 char name[20],name2[20];
 tel t1;
 count=0;
 fstream g,f;
 do
 {
  cout<<"\n>>>>>>>>>>>>>>>>>>>>>>>MENU<<<<<<<<<<<<<<<<<<<<";
  cout<<"\n1.Insert and overwrite\n2.Show\n3.Search & Edit(number)\n4.Search &
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();
```

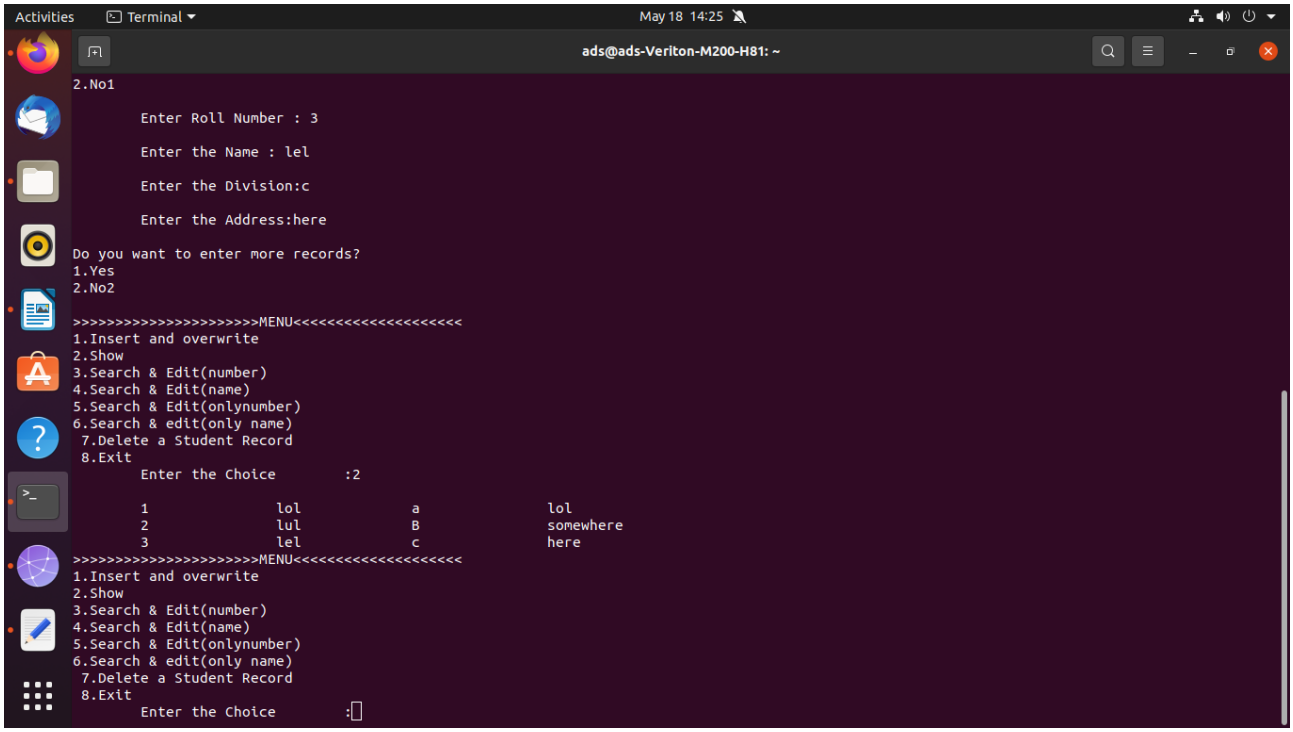
```
  n1=(add-start)/(sizeof(t1));
  f.seekp((n1-1)*sizeof(t1),ios::beg);
  t1.accept();
  f.write((char*) &t1,(sizeof(t1)));
  f.close();
  count++;
  break;
 }
 f.read((char*)&t1,(sizeof(t1)));
    }
if(count==0)
      cout<<"\nRecord not found";
f.close();
break;

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();
```

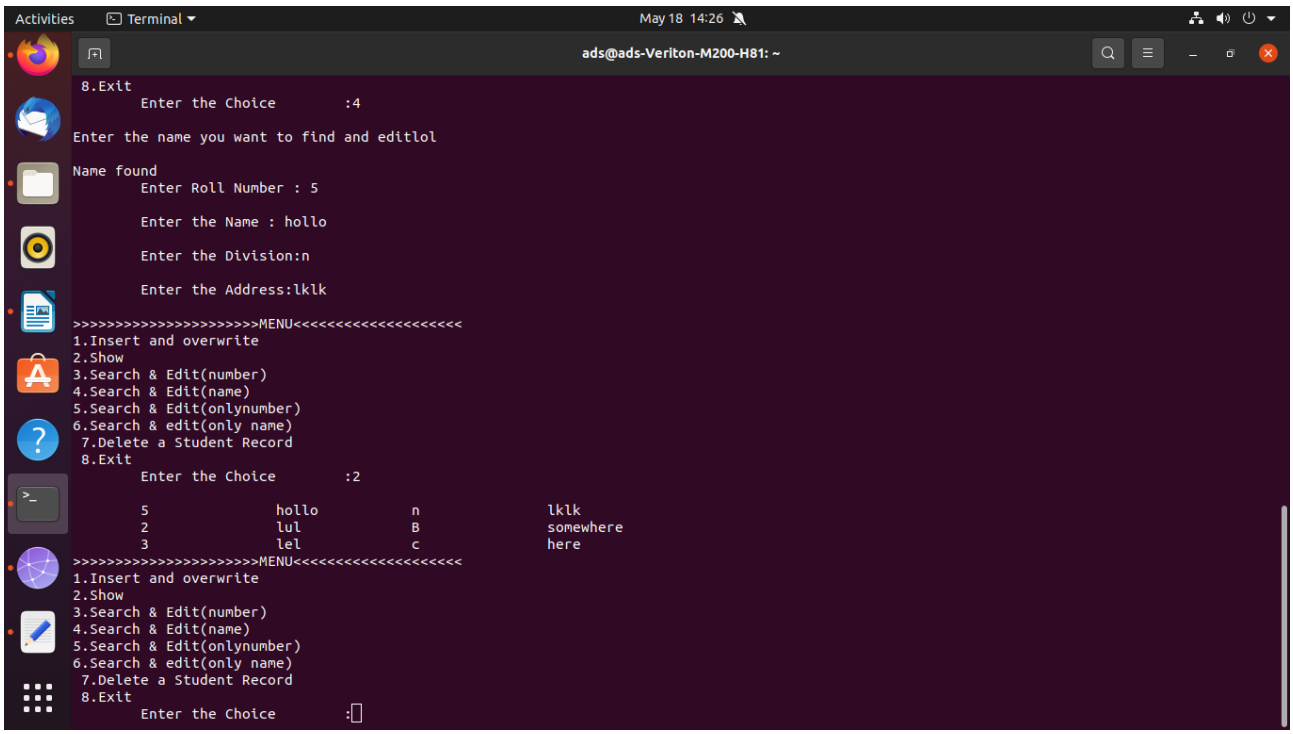
```
            n3=(add3-start3)/(sizeof(t1));
            f.seekp((n3-1)*(sizeof(t1)),ios::beg);
            t1.accept2();
            f.write((char*)&t1,(sizeof(t1)));
            f.close();
            break;
           }
          f.read((char*)&t1,(sizeof(t1)));
        }
        break;
   case 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");
```

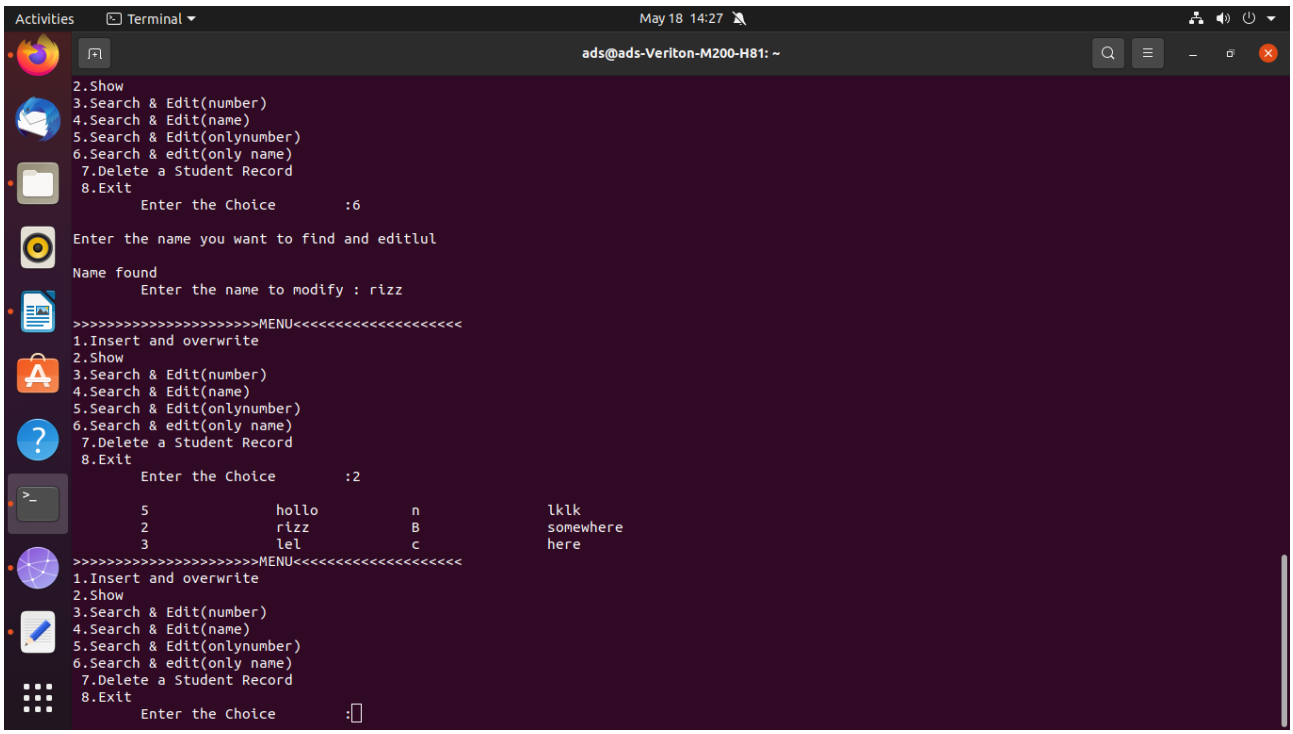
```
      break;
    case 8:
      cout<<"\n\tThank you";
      break;


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

Output:


Activities
Terminal
May 18 14:25
ads@ads-Veriton-M200-H81: ~
>>>>>>>>>>>>>>>>>>>>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 : 1
Enter the Name : lol
Enter the Division:a
Enter the Address:lol
Do you want to enter more records?
1.Yes
2.No1
Enter Roll Number : 2
Enter the Name : lul
Enter the Division:B
Enter the Address:somewhere
Do you want to enter more records?
1.Yes
2.No1
Enter Roll Number : 3


Activities
Terminal
May 18 14:25
ads@ads-Veriton-M200-H81: ~
2.No1
Enter Roll Number : 3
Enter the Name : lel
Enter the Division:c
Enter the Address:here
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
1 lol a lol
2 lul B somewhere
3 lel c here
>>>>>>>>>>>>>>>>>>>>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 :


Activities
Terminal
May 18 14:26
ads@ads-Veriton-M200-H81: ~
8.Exit
Enter the Choice :4
Enter the name you want to find and editlol
Name found
Enter Roll Number : 5
Enter the Name : hollo
Enter the Division:n
Enter the Address:lklk
>>>>>>>>>>>>>>>>>>>>>>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
5 hollo n lklk
2 lul B somewhere
3 lel c here
>>>>>>>>>>>>>>>>>>>>>>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 :


Activities
Terminal
May 18 14:27
ads@ads-Veriton-M200-H81: ~
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 :6
Enter the name you want to find and editlul
Name found
Enter the name to modify : rizz
>>>>>>>>>>>>>>>>>>>>>>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
5 hollo n lklk
2 rizz B somewhere
3 lel c here
>>>>>>>>>>>>>>>>>>>>>>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 :

Experiment No: 13

Implement the Heap/Shell sort algorithm implemented in Java demonstrating heap/shell data structure with modularity of programming language.

```
import java.util.*;
public class dsa13 {

 private static int N;
   public static void sort(int arr[]){
  heapMethod(arr);
      for (int i = N; i > 0; i--){
         swap(arr,0, i);
         N = N-1;
         heap(arr, 0);
      }
   }
   public static void heapMethod(int arr[]){
      N = arr.length-1;
      for (int i = N/2; i >= 0; i--)
         heap(arr, i);
   }
   public static void heap(int arr[], int i){
      int left = 2*i ;
      int right = 2*i + 1;
      int max = i;
      if (left <= N && arr[left] > arr[i])
         max = left;
  if (right <= N && arr[right] > arr[max])
         max = right;
      if (max != i){
         swap(arr, i, max);
         heap(arr, max);
      }
   }
   public static void swap(int arr[], int i, int j){
      int tmp = arr[i];
      arr[i] = arr[j];
      arr[j] = tmp;
   }
   public static void main(String[] args) {
      Scanner in = new Scanner( System.in );
      int n;
      System.out.println("Enter the number of elements to be sorted:");
      n = in.nextInt();
      int arr[] = new int[ n ];
      System.out.println("Enter "+ n +" integer elements");
      for (int i = 0; i < n; i++)
         arr[i] = in.nextInt();
      sort(arr);
      System.out.println("After sorting ");
      for (int i = 0; i < n; i++)
```

```
            System.out.println(arr[i]+" ");
        System.out.println();
    }

}
```

Output: