ASSIGNMENT NO -1

AIM:
To create ADT that implement the "set" concept.
a. Add (new Element) -Place a value into the set
b. Remove (element)
c. Contains (element) Return true if element is in collection
d. Size () Return number of values in collection
e. Intersection of two sets
f. Union of two sets
g. Difference between two sets
h. Subset

CODE:
```
#include<iostream>
#include<stdlib.h>
using namespace std;

void create(int set[])
{
int n;
cout<<"\n enter the size of set : ";
cin>>n;
cout<<"\n enter the elements in the set : ";
for(inti=1;i<=n;i++)
cin>>set[i];

set[0]=n;
}

bool member(int set[],intnum)
{
for(inti=1;i<=set[0];i++)
if(set[i]==num)
return true;

return false;
}
void intersection(int set1[],int set2[],int set3[])
{
for(inti=1;i<=set2[0];i++)
  {
if(member(set1,set2[i])== true)
    {
```

```
set3[0]++;
set3[set3[0]]=set2[i];
    }

  }
}
void union1(int set1[],int set2[],int set4[])
{

for(inti=0;i<=set1[0];i++)
set4[i]=set1[i];

for(inti=1;i<=set2[0];i++)
  {
if(member(set1,set2[i])== false)
    {
set4[0]++;
set4[set4[0]]=set2[i];
    }
  }
}

void difference1(int set1[],int set2[],int set5[])
{
for(inti=1;i<=set1[0];i++)
  {
if((member(set2,set1[i]) == false))
    {
set5[0]++;
set5[set5[0]]=set1[i];
    }
 ;  }
}

void contains(int set[])
{
intnum;
cout<<"\n enter the element to be searched ";
cin>>num;
if((member(set,num))== true)
cout<<"\n element is present ";
else
cout<<"\n element is not present ";
}
```

```
bool subset(int seta[],intsetb[])
{
for(inti=1;i<=setb[0];i++)
    {
if((member(seta,setb[i]))==true)
continue;
else
return false;
    }
return true;
}

void remove(int set[])
{
intpos;
cout<<"\n enter the position from which you want to remove the element : ";
cin>>pos;
if(pos<=set[0])
    {
if(pos<set[0])
        {
for(inti=pos;i<=set[0];i++)
            {
set[i]=set[i+1];
            }
set[0]--;
        }
else if(pos==set[0])
        {
set[0]--;
        }
    }
else
    {
cout<<"\n entered position exceeds the size of the set " ;
    }
}

void size(int set[])
{
cout<<set[0];

}
```

```cpp
void display(int set[])
{
cout<<"\n size : "<<set[0]<<"\t";
for(inti=1;i<=set[0];i++)
    {
cout<<set[i]<<"   ";
    }
}




int main()
{
int set1[10];
cout<<"\n FOR SET 1 ";
create(set1);

int set2[10];
cout<<"\n FOR SET 2 ";
create(set2);

intch,c;
char choice;
do{
cout<<"\n\n -------------- OPERATION MENU --------------- ";
cout<<"\n 1 for INTERSECTION ";
cout<<"\n 2 for UNION ";
cout<<"\n 3 for DIFFERENCE ";
cout<<"\n 4 for CONTAINS( if element is present in set or not)";
cout<<"\n 5 for SUBSET";
cout<<"\n 6 for REMOVE";
cout<<"\n 7 for SIZE";
cout<<"\n 8 for DISPLAY";
cout<<"\n 9 for EXIT";
cout<<"\n\n Enter your choice : ";
cin>>ch;
switch(ch)
{
case 1:
     {
int set3[1];
set3[0]=0;
```

```
cout<<"\n the intersection of two sets : \t";
intersection(set1,set2,set3);
display(set3);
break;
     }
case 2:
     {
int set4[set1[0]+1];
set4[0]=0;
cout<<"\n the union of two sets \t";
union1(set1,set2,set4);
display(set4);
break;
     }
case 3:
     {
int set5[1];
set5[0]=0;
cout<<"\n the difference of two sets \t";
difference1(set1,set2,set5);
display(set5);
break;
     }
case 4:
     {
cout<<"\n enter 1 for searching in set1 and 2 for searching in set2 ";
cin>>c;
switch(c)
        {
case 1: contains(set1); break;
case 2: contains(set2); break;
default: cout<<"\n wrong choice entered ";
        }

break;
     }

case 5:
     {
label:
cout<<"\n enter 1 for checking if set1 is subset of set2 else enter 2 ";
cin>>c;
switch(c)
        {
```

```
case 1:
        {
if(subset(set2,set1)==true )
cout<<"\n set1 is subset of set2";
else
cout<<"\n set1 is not a subset of set2";
break;
        }
case 2:
        {
if(subset(set1,set2)==true )
cout<<"\n set2 is subset of set1";
else
cout<<"\n set2 is not a subset of set1";
break;
        }
default:
cout<<"\n wrong choice entered ";
goto label;
        }
break;
    }
case 6:
    {
        label2:
cout<<"\n enter 1 for removing element from set 1 and 2 for removal from set
2 ";
cin>>c;
switch(c)
        {
case 1:
        {
remove(set1);
break;
        }
case 2:
        {
remove(set2);
break;
        }
default:
cout<<"\n wrong choice entered ";
goto label2;
        }
```

NAME: RUTUJA KAUTHALE, ROLL: 221058, BATCH: A3, GR NO. : 17U251

```
break;
        }
case 7:
        {
cout<<"\n size of set 1 : ";
size(set1);
cout<<"\n size of set 2 : ";
size(set2);
break;
        }
case 8:
        {
display(set1);
display(set2);
break;
        }

case 9:
exit(0);
default:
cout<<"\n wrong choice entered ";
}

cout<<"\n want to continue with the operation ?(y/n) :";
cin>>choice;

}while((choice=='y')||(choice=='Y'));

return 0;
}
```

OUTPUT:

```
FOR SET 1
enter the size of set : 4

enter the elements in the set : 1 2 3 4

FOR SET 2
enter the size of set : 3

enter the elements in the set : 3 4 5


-------------- OPERATION MENU ----------------
1 for INTERSECTION
2 for UNION
3 for DIFFERENCE
4 for CONTAINS( if element is present in set or not)
5 for SUBSET
6 for REMOVE
7 for SIZE
8 for DISPLAY
9 for EXIT

Enter your choice : 1

the intersection of two sets :
size : 2        3     4
want to continue with the operation ?(y/n) :Y


-------------- OPERATION MENU ----------------
1 for INTERSECTION
2 for UNION
3 for DIFFERENCE
4 for CONTAINS( if element is present in set or not)
5 for SUBSET
6 for REMOVE
7 for SIZE
8 for DISPLAY
9 for EXIT

Enter your choice : 2
```

```
Enter your choice : 2

the union of two sets
size : 5        1    2    3    4    5
want to continue with the operation ?(y/n) :Y


-------------- OPERATION MENU ----------------
1 for INTERSECTION
2 for UNION
3 for DIFFERENCE
4 for CONTAINS( if element is present in set or not)
5 for SUBSET
6 for REMOVE
7 for SIZE
8 for DISPLAY
9 for EXIT

Enter your choice : 3

the difference of two sets
size : 2        1    2
want to continue with the operation ?(y/n) :Y

-------------- OPERATION MENU ----------------
1 for INTERSECTION
2 for UNION
3 for DIFFERENCE
4 for CONTAINS( if element is present in set or not)
5 for SUBSET
6 for REMOVE
7 for SIZE
8 for DISPLAY
9 for EXIT

Enter your choice : 4

enter 1 for searching in set1 and 2 for searching in set2 1

enter the element to be searched 2

element is present
want to continue with the operation ?(y/n) :
```

```
Enter your choice : 4

enter 1 for searching in set1 and 2 for searching in set2 1

enter the element to be searched 2

element is present
want to continue with the operation ?(y/n) :Y

-------------- OPERATION MENU ----------------
1 for INTERSECTION
2 for UNION
3 for DIFFERENCE
4 for CONTAINS( if element is present in set or not)
5 for SUBSET
6 for REMOVE
7 for SIZE
8 for DISPLAY
9 for EXIT

Enter your choice : 5

enter 1 for checking if set1 is subset of set2 else enter 2 1

set1 is not a subset of set2
want to continue with the operation ?(y/n) :Y

-------------- OPERATION MENU ----------------
1 for INTERSECTION
2 for UNION
3 for DIFFERENCE
4 for CONTAINS( if element is present in set or not)
5 for SUBSET
6 for REMOVE
7 for SIZE
8 for DISPLAY
9 for EXIT

Enter your choice : 6

enter 1 for removing element from set 1 and 2 for removal from set 2
```

```
set1 is not a subset of set2
want to continue with the operation ?(y/n) :Y

-------------- OPERATION MENU ----------------
1 for INTERSECTION
2 for UNION
3 for DIFFERENCE
4 for CONTAINS( if element is present in set or not)
5 for SUBSET
6 for REMOVE
7 for SIZE
8 for DISPLAY
9 for EXIT

Enter your choice : 6

enter 1 for removing element from set 1 and 2 for removal from set 2 1

enter the position from which you want to remove the element : 2

want to continue with the operation ?(y/n) :Y

-------------- OPERATION MENU ----------------
1 for INTERSECTION
2 for UNION
3 for DIFFERENCE
4 for CONTAINS( if element is present in set or not)
5 for SUBSET
6 for REMOVE
7 for SIZE
8 for DISPLAY
9 for EXIT

Enter your choice : 8

size : 3      1    3    4
size : 3      3    4    5
want to continue with the operation ?(y/n) :7

------------------------------
Process exited after 261.4 seconds with return value 0
Press any key to continue . . .
```

NAME: RUTUJA KAUTHALE, ROLL: 221058, BATCH: A3, GR NO. : 17U251

ASSIGNMENT NO- 2

AIM:
Construct a threaded binary search tree by inserting values in the given order and traverse it in inorder traversal using threads.

CODE:

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

class ttree
{
        private:
                struct thtree
                {
                        int left;
                        thtree *leftchild;
                        int data;
                        thtree *rightchild;
                        int right;
                }*th_head;

        public:
                ttree();
                void insert(int num);
                void inorder();

};

ttree::ttree()
{
        th_head=NULL;
}
void ttree::insert(int num)
{


        thtree *head=th_head,*p,*z;

        z=new thtree;
        z->left=true;
        z->data=num;
        z->right=true;
```

```
if(th_head==NULL)
{
        head=new thtree;
        head->left=false;
        head->leftchild=z;
        head->data=-9999;
        head->rightchild=head;
        head->right=false;

        th_head=head;
        z->leftchild=head;
        z->rightchild=head;

}
else
{
        p=head->leftchild;
        while(p!=head)
        {
                if(p->data > num)
                {
                        if(p->left!=true)
                        p=p->leftchild;
                        else
                        {
                                z->leftchild=p->leftchild;
                                p->leftchild=z;

                                p->left=false;
                                z->right=true;
                                z->rightchild=p;
                                return;
                        }
                }
                else
                {
                        if(p->data < num)
                        {
                                if(p->right!=true)
                                p=p->rightchild;
                                else
                                {
                                        z->rightchild=p->rightchild;
                                        p->rightchild=z;

                                        p->right=false;
```

```
                                        z->left=true;
                                        z->leftchild=p;
                                        return;
                                    }
                                }
                            }
                        }
                    }
}
void ttree::inorder()
{
        thtree *a;
        a=th_head->leftchild;
        while(a!=th_head)
        {
                while(a->left==false)

                        a=a->leftchild;
                        cout<<a->data<<"\t";



                while(a->right==true)
                {
                        a=a->rightchild;

                        if(a==th_head)
                        break;

                        cout<<a->data<<"\t";
                }
                a=a->rightchild;
        }
}

int main()
{
        ttree th;
        int n,e;
        cout<<"Enter no. of elements: ";
        cin>>n;
        cout<<"\nEnter elements: ";
        for(int i=0;i<n;i++)
        {
                cin>>e;
                th.insert(e);
```
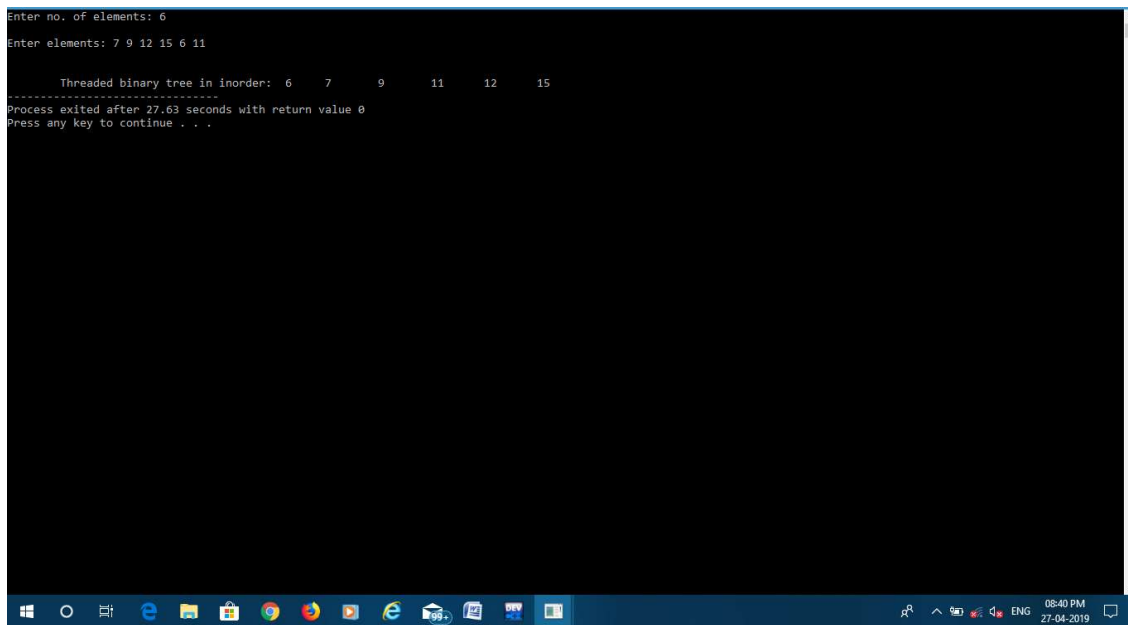
```
        }

    cout<<"\n\n\tThreaded binary tree in inorder:  ";
    th.inorder();
}
```

OUTPUT:

```
Enter no. of elements: 6

Enter elements: 7 9 12 15 6 11

        Threaded binary tree in inorder:  6     7      9      11     12     15
--------------------------------
Process exited after 27.63 seconds with return value 0
Press any key to continue . . .
```

ASSIGNMENT NO- 3

AIM:
 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 takes 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. Justify the storage representations
used.

CODE:
```
#include<iostream>
#define MAX 20
using namespace std;
class dijkstra
{
int city;
int distance[MAX][MAX];
int d[MAX];
int visited[MAX];
public:
void city_no();
int minvertex();
void matrix_fill();
void dijkstra_code();
void display();
};
void dijkstra::city_no()
{
```
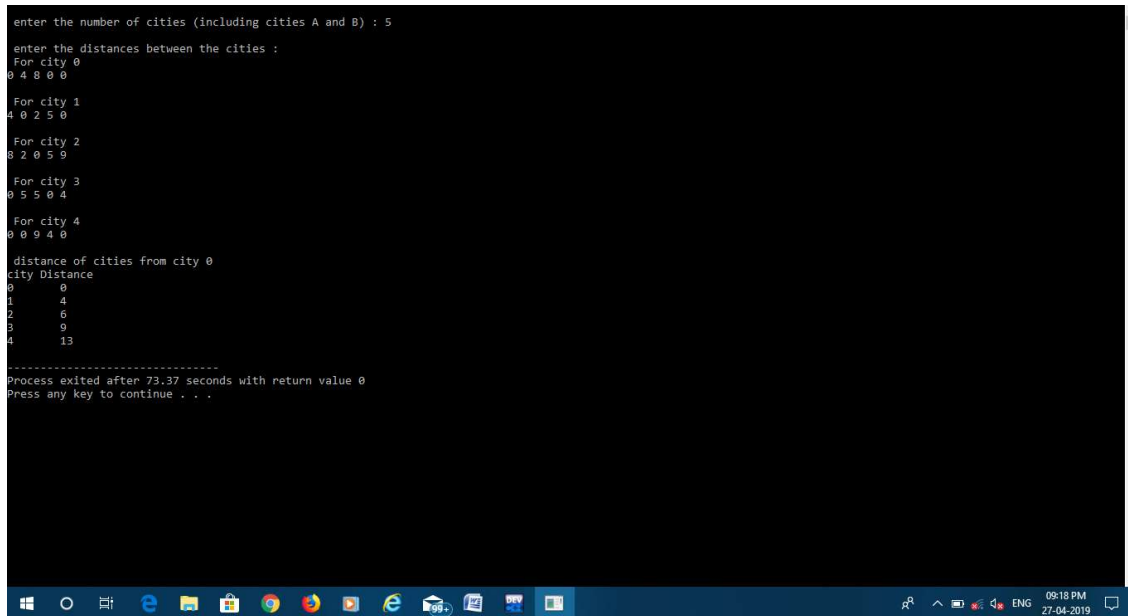
```cpp
cout<<"\n enter the number of cities (including cities A and B) : ";
cin>>city;
}
int dijkstra::minvertex()
{
int mvertex=-1;
for(int i=0;i<city;i++)
{
if(visited[i]==0 && (mvertex==-1 || d[i]<d[mvertex]))
mvertex=i;
}
return mvertex;
}
void dijkstra::matrix_fill()
{
cout<<"\n enter the distances between the cities : ";
for(int i=0;i<city;i++)
{
cout<<"\n For city "<<i<<endl;
for(int j=0;j<city;j++)
{
if(i==j)
distance[i][j]=0;
cin>>distance[i][j];
}
d[i]=INT_MAX;
visited[i]=0;
}
}
void dijkstra::dijkstra_code()
{
d[0]=0;
for(int i=0;i<city-1;i++)
{
int mvertex=minvertex();
visited[mvertex]=1;
for(int j=0;j<city;j++)
{
if((distance[mvertex][j]!=0)&&(visited[j]==0))
{
int dist=d[mvertex]+distance[mvertex][j];
if(dist<d[j])
d[j]=dist;
}
}
}
}
```

```
}
void dijkstra::display()
{
cout<<"\n distance of cities from city 0 \n";
cout<<"city Distance\n";
for(int i=0;i<city;i++)
cout<<i<<"\t"<<d[i]<<endl;
}
int main()
{
dijkstra sp;
sp.city_no();
sp.matrix_fill();
sp.dijkstra_code();
sp.display();
return 0;
}
```

OUTPUT:

ASSIGNMENT NO-4

AIM:

For a weighted graph G, find the minimum spanning tree using Prims Algorithm.

CODE:

```
#include<iostream>

using namespace std;

void create(int mat[][10], int v)
{
int v1,v2,cost;
int edges;
cout<< "\nEnter the total number of edges : ";
cin>> edges;

for(inti=0;i<v;i++)
        {
                for(int j=0;j<v;j++)
                {
                        mat[i][j] = 0;
                }
        }

for(inti=0;i<edges;i++)
   {
cout<< "\nEnter edge : ";
cin>> v1 >> v2;
cout<< "\nEnter the cost of that edge : ";
cin>> cost;
mat[v1][v2] = cost;
   }
}

voiddisplay_matrix(int mat[][10],int v)
{
cout<< "\nAdjacency matrix representation :- " <<endl;
for(inti=0;i<v;i++)
   {
for(int j=0;j<v;j++)
      {
cout<< mat[i][j] << "\t";
      }
cout<<endl;
   }
```

```cpp
}

intmin_dist(intdist[],int visited[],int v)
{
int min = 32767;
intmin_index;
for(inti=0;i<v;i++)
    {
if(visited[i] == 0 &&dist[i] <= min)
        {
min = dist[i];
min_index = i;
        }
    }
cout<<min_index<<endl;
returnmin_index;
}

voidprint_dist(int mat[][10],intdist[],intv,int parent[])
{
int sum=0;
cout<<"\nPRIM'S MST OF THE GRAPH IS: ";
for(inti = 1; i<v; i++)
    {
cout<<"\n"<<i<<"-"<<parent[i];
sum = sum + mat[parent[i]][i];
    }
cout<<endl;
cout<<"\nCOST OF MST IS: "<<sum<<endl;
}

void prims(int mat[][10],ints,int v)
{
intdist[v];
int visited[v];
int parent[v];
for(inti=0;i<v;i++)
    {
dist[i] = 32767;
visited[i] = 0;
    }
dist[s] = 0;
int p=0;
for(int j=0;j<v-1;j++)
    {
        p = min_dist(dist,visited,v);
```

```
visited[p] = 1;
for(int q=0;q<v;q++)
        {
if(mat[p][q] != 0)
          {
if(visited[q] == 0 && mat[p][q] <dist[q])
            {
dist[q] = mat[p][q];
parent[q] = p;
            }
          }
        }
    }
print_dist(mat,dist,v,parent);
}




int main()
{
int v;
int s;
cout<< "\nEnter the number of vertices : ";
cin>> v;
int mat[v][10];
        create(mat,v);
display_matrix(mat,v);
cout<< "\nEnter source vertex : ";
cin>> s;
prims(mat,s,v);
return 0;
}
```

OUTPUT:

VIIT PUNE, BRANCH: COMPUTER SCIENCE, BATCH:2017(PATTERN)

```
Enter the number of vertices : 3

Enter the total number of edges : 2

Enter edge : 0 1

Enter the cost of that edge : 5

Enter edge : 1 2

Enter the cost of that edge : 10

Adjacency matrix representation :-
0      5      0
0      0      10
0      0      0

Enter source vertex : 0 0 1
0
1

PRIM'S MST OF THE GRAPH IS:
1-0
2-1

COST OF MST IS: 15

-------------------------------
Process exited after 77.11 seconds with return value 0
Press any key to continue . . .
```

NAME: RUTUJA KAUTHALE, ROLL: 221058, BATCH: A3, GR NO. : 17U251

ASSIGNMENT NO- 5

AIM:

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

CODE:

```cpp
include<iostream>

using namespace std;

#define MAX 30

typedef struct edge
{
    int u,v,w;
}edge;

typedef struct edgelist
{
    edge data[MAX];
    int count;
}edgelist;

edgelist elist;

int G[MAX][MAX],n;

edgelist spanlist;


void kruskal();

int find(int belongs[],int vertexno);

void union1(int belongs[],int c1,int c2);
```

```cpp
void sort();

void print();

int main()

{

    int i,j;

    cout<<"\nEnter number of city's:";

    cin>>n;

cout<<"\nEnter the adjacency matrix of city ID's:\n";

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

    for(j=0;j<n;j++)

        cin>>G[i][j];

    kruskal();

    print();

}

void kruskal()

{

    int belongs[MAX],i,j,cno1,cno2;

    elist.count=0;

    for(i=1;i<n;i++)

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

        {

            if(G[i][j]!=0)

            {

                elist.data[elist.count].u=i;

                elist.data[elist.count].v=j;
```

```
            elist.data[elist.count].w=G[i][j];

            elist.count++;

        }

    }

    sort();

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

        belongs[i]=i;

    spanlist.count=0;

    for(i=0;i<elist.count;i++)

    {

        cno1=find(belongs,elist.data[i].u);

        cno2=find(belongs,elist.data[i].v);


        if(cno1!=cno2)

        {

            spanlist.data[spanlist.count]=elist.data[i];

            spanlist.count=spanlist.count+1;

            union1(belongs,cno1,cno2);

        }

    }

}

int find(int belongs[],int vertexno)

{

    return(belongs[vertexno]);

}
```

```
void union1(int belongs[],int c1,int c2)

{

    int i;

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

        if(belongs[i]==c2)

            belongs[i]=c1;

}

void sort()

{

    int i,j;

    edge temp;

    for(i=1;i<elist.count;i++)

        for(j=0;j<elist.count-1;j++)

            if(elist.data[j].w>elist.data[j+1].w)

            {

                temp=elist.data[j];

                elist.data[j]=elist.data[j+1];

                elist.data[j+1]=temp;

            }

}

void print()

{

    int i,cost=0;

    for(i=0;i<spanlist.count;i++)

    {
```
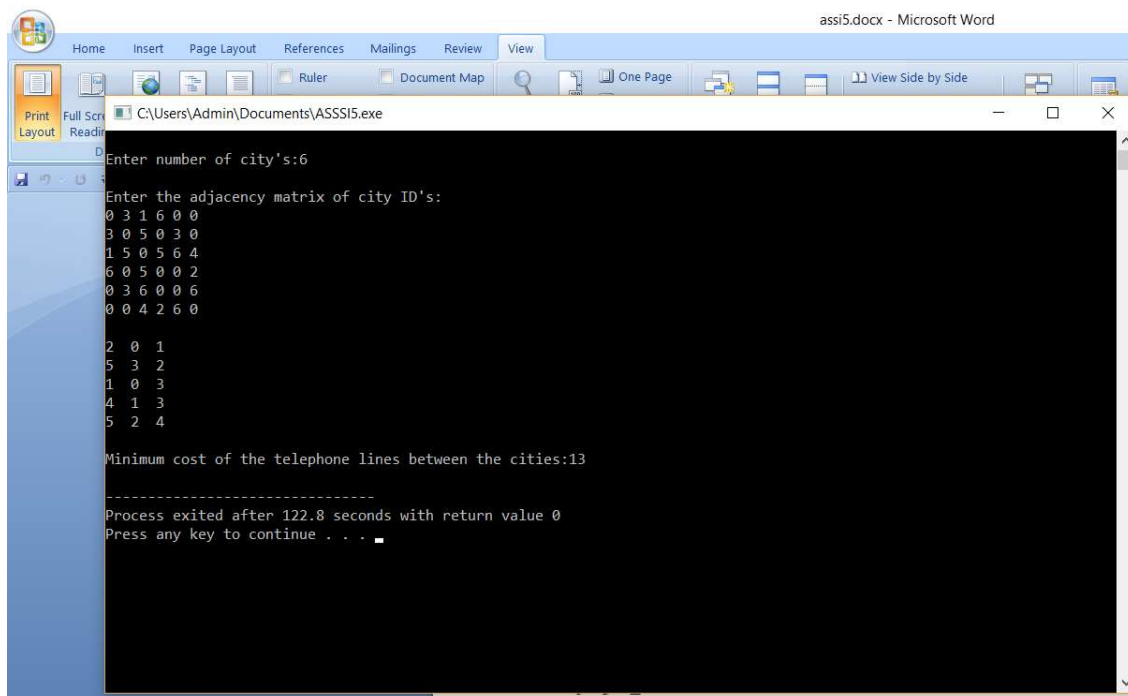
```
    cout<<"\n"<<spanlist.data[i].u<<"  "<<spanlist.data[i].v<<"  "<<spanlist.data[i].w;

    cost=cost+spanlist.data[i].w;

}


    cout<<"\n\nMinimum cost of the telephone lines between the cities:"<<cost<<"\n";

}
```

OUTPUT:

ASSIGNMENT NO- 6

AIM:

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

CODE:

```cpp
#include <iostream>

using namespace std;

// Toheapify a subtree rooted with node i which is

// an index in arr[]. n is size of heap

voidheapify(intarr[], int n, inti)

{
        int largest = i; // Initialize largest as root

        int l = 2*i + 1; // left = 2*i + 1

        int r = 2*i + 2; // right = 2*i + 2

        // If left child is larger than root

        if (l < n &&arr[l] >arr[largest])

                largest = l;

        // If right child is larger than largest so far

        if (r < n &&arr[r] >arr[largest])

                largest = r;

        // If largest is not root

        if (largest != i)

        {
                swap(arr[i], arr[largest]);

                // Recursively heapify the affected sub-tree
```

```
        heapify(arr, n, largest);

    }

}
// main function to do heap sort
voidheapSort(intarr[], int n)

{

    for (inti = n / 2 - 1; i>= 0; i--)

        heapify(arr, n, i);

    for (inti=n-1; i>=0; i--)

    {

        swap(arr[0], arr[i]);

        heapify(arr, i, 0);

    }

}
int main()

{

intn,arr[100];

    cout<<"Enter the no. of student's marks you want to enter. :\n";

    cin>>n;

    cout<<"Enter the marks :\n";

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

  {

cin>>arr[i];

  }

    heapSort(arr, n);
```
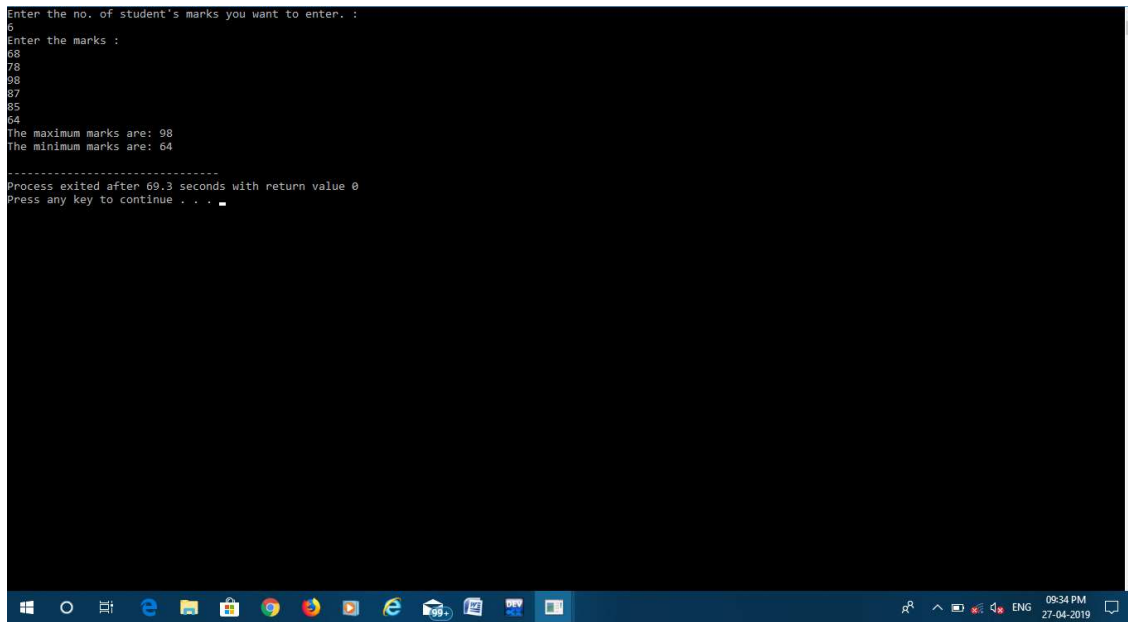
```
cout<< "The maximum marks are: "<<arr[n-1]<<"\n";

cout<<"The minimum marks are: "<<arr[0]<<"\n";

}
```

OUTPUT:

```
Enter the no. of student's marks you want to enter. :
6
Enter the marks :
68
78
98
87
85
64
The maximum marks are: 98
The minimum marks are: 64

------------------------------
Process exited after 69.3 seconds with return value 0
Press any key to continue . . .
```

ASSIGNMENT NO- 7

AIM:

Insert the keys into a hash table of length m using open addressing using double hashing with h(k)=1+(k mod (m-1)).

CODE:

```
#include<iostream>

#include<stdlib.h>

using namespace std;


int size=10;

void display(int hash[])

{

        int i;

        cout<<"\nHASH TABLE:\n";

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

        {

                cout<<""<<hash[i]<<"\t";

        }

}

int main()

{

        int hash[size],val,i;

        char ch;

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

        {
```

```
        hash[i]=-1;
}
do
{
        cout<<"Enter the value:  ";
        cin>>val;
        int m=val%size;
        if(hash[m] == -1)
        {
                hash[m]=val;
                display(hash);
        }
else
{
        cout<<"\nCollision: ";
        cout<<""<<hash[m]<<"\n";
        if(hash[m]%10!=m)
        {
                int h=hash[m];
                hash[m]=val;
                val=h;
        }
        int x=1+(val%(size-1));
        for(i=1;i<size;i++)
        {
```

```
                    int y=(val+i*x);

                    int z=y%size;

                    if(hash[z]==-1)

                    {

                            hash[z]=val;

                            display(hash);

                            break;

                    }

                }

        }

        cout<<"\n\n Want to add more values?  ";

        cin>>ch;

        int ss=0;

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

        {

                if(hash[i]!= -1)

                ss++;

        }

        if(ss==10)

        {

                cout<<"\n\nHast table is full";

                exit(1);

        }

}while(ch=='y' || ch=='Y');

display(hash);
```

return 0;

}

OUTPUT:

```
Enter the value: 11

HASH TABLE:
-1      11      -1      -1      -1      -1      -1      -1      -1      -1

 Want to add more values?  y
Enter the value:  32

HASH TABLE:
-1      11      32      -1      -1      -1      -1      -1      -1      -1

 Want to add more values?  y
Enter the value:  64

HASH TABLE:
-1      11      32      -1      64      -1      -1      -1      -1      -1

 Want to add more values?  y
Enter the value:  74

Collision: 64

HASH TABLE:
-1      11      32      -1      64      -1      -1      74      -1      -1

 Want to add more values?  y
Enter the value:  66

HASH TABLE:
-1      11      32      -1      64      -1      66      74      -1      -1

 Want to add more values?  y
Enter the value:  97

Collision: 74

HASH TABLE:
74      11      32      -1      64      -1      66      97      -1      -1

 Want to add more values?  n

HASH TABLE:
74      11      32      -1      64      -1      66      97      -1      -1
```

ASSIGNMENT NO- 8

AIM:

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 maintain data

CODE:

```
#include<iostream>

#include<fstream>

using namespace std;

class student

{

        int roll_num;

        char div;

        string name;

        string address;

        public:

        void getdata()

        {

                cout<<"\n Enter the Roll Number: ";

                cin>>roll_num;

                cout<<"Enter the division: ";

                cin>>div;

                cout<<"Enter the Name: ";
```

```
        fflush(stdin);

        getline(cin,name);

        cout<<"Enter the Address: ";

        fflush(stdin);

        getline(cin,address);

}

void putdata(int n)

{

        student st[n];

        ifstream infile;

        infile.open("student.dat",ios::binary|ios::in);

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

   {

                infile.read((char *)&st[i],sizeof(st[i]));

                cout<<"\n Roll Number: "<<st[i].roll_num;

                cout<<"\n Division:  "<<st[i].div;

                fflush(stdin);

                cout<<"\n Name:  "<<st[i].name;

                fflush(stdin);

                cout<<"\n Address:  "<<st[i].address;

                cout<<"\n
_____ \n";

   }

        infile.close();

}
```

```
void search_(int n)

{

        student st[n];

        ifstream infile;

        cout<<"\n Enter the Roll Number to be searched: ";

        int r;

        cin>>r;

        infile.open("student.dat",ios::in|ios::binary);

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

        {

                infile.read((char *)&st[i],sizeof(st[i]));

                if(st[i].roll_num==r)

                {

                        cout<<"\n Found";

                        cout<<"\n Details:  "<<endl;

                        cout<<"\n Roll Number: "<<st[i].roll_num;

                        cout<<"\n Division:  "<<st[i].div;

                        fflush(stdin);

                        cout<<"\n Name:  "<<st[i].name;

                        fflush(stdin);

                        cout<<"\n Address:  "<<st[i].address;

                        cout<<"\n
_____ \n";
```

```cpp
                    infile.close();

                    return;

        }

    }

    cout<<"\n Not Found";

    infile.close();

}


void del(int n)

{

    student st[n];

    int r;

    cout<<"\n Enter the roll number to be deleted ";

    cin>>r;

    ifstream infile;

    ofstream outfile;

    infile.open("student.dat",ios::binary|ios::in);

    outfile.open("temp.dat",ios::binary|ios::out);

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

    {

        infile.read((char *)&st[i],sizeof(st[i]));

        if(st[i].roll_num==r)

        {

            continue;

        }
```

```
                else
                {
                        outfile.write((char *)&st[i],sizeof(st[i]));
                }
        }
        outfile.close();
        infile.close();
        remove("student.dat");
        int re=rename("temp.dat","student.dat");
        cout<<"Data deleted";


    }
};
int main()
{
        int n;
        cout<<"\nEnter the Number of Students:   ";
        cin>>n;
        student s[n];
        ofstream outfile;
        outfile.open("student.dat",ios::out|ios::binary);
        for(int i=0;i<n;i++)
        {
                cout<<"\n Enter the information of Students: ";
                s[i].getdata();
```

```cpp
        outfile.write((char *)&s[i],sizeof(s[i]));

}

outfile.close();


int c;

student d;

do

{

        cout<<"\n 1.Search";

        cout<<"\n 2.Delete";

        cout<<"\n 3.Display";

        cout<<"\n 4.Exit";

        cout<<"\n Enter Your Choice";

        cin>>c;

        switch(c)

        {

                case 1:d.search_(n);break;

                case 2:d.del(n);n=n-1;break;

                case 3:d.putdata(n);break;

                case 4:break;

        }

}

while(c!=4);



}
```

NAME: RUTUJA KAUTHALE, ROLL: 221058, BATCH: A3, GR NO. : 17U251

OUTPUT:

```
Enter the Number of Students:   3

 Enter the information of Students:
 Enter the Roll Number: 1
Enter the division: A
Enter the Name: Akanksha
Enter the Address: Pune

 Enter the information of Students:
 Enter the Roll Number: 2
Enter the division: B
Enter the Name: Shraddha
Enter the Address: Parbhani

 Enter the information of Students:
 Enter the Roll Number: 3
Enter the division: C
Enter the Name: Neha
Enter the Address: Akola

1.Search
2.Delete
3.Display
4.Exit
Enter Your Choice1

Enter the Roll Number to be searched: 1

 Found
Details:

Roll Number: 1
Division:  A
Name:  Akanksha
Address:  Pune
_____

1.Search
2.Delete
3.Display
4.Exit
Enter Your Choice
```

ASSIGNMENT NO- 9

AIM:

Department maintains a employee information. The file contains 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 main the data.

Title:

Index Sequential File

Problem Statement:

Department maintains a employee information. The file contains 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 main the data.

OBJECTIVE:

To make use of index sequential files to maintain and operation on data.

SOFTWARE AND HARDWARE REQUIREMENT:

1. 64-bit Open source Linux or itsderivative.

2. Open Source C++ Programming tool like G++/GCC.

THEORY:

INDEX SEQUENTIAL FILE:

This is basically a mixture of sequential and indexed file organisation techniques. Records are held in sequential order and can be accessed randomly through an index. Thus, these files share the merits of both systems enabling sequential or direct access to the data.
The index to these files operates by storing the highest record key in given cylinders and tracks. Note how this organisation gives the index a tree structure. Obviously this type of file organisation will require a direct access device, such as a hard disk.

Indexed sequential file organisation is very useful where records are often retrieved randomly and are also processed in (sequential) key order. Banks may use this organisation for their auto-bank machines i.e. customers randomly access their accounts throughout the day and at the end of the day the banks can update the whole file sequentially.

Advantages of Indexed Sequential Files:

1.Allows records to be accessed directly or sequentially.

2.Direct access ability provides vastly superior (average) access times.

Disadvantages of Indexed Sequential Files:

1.The fact that several tables must be stored for the index makes for a considerable

storage overhead.

2.As the items are stored in a sequential fashion this adds complexity to the

addition/deletion of records. Because frequent updating can be very inefficient,

especially for large files, batch updates are often performed.

CONCLUSION:

In above assignment, we made the use of index sequential files to operate on employee data.

NAME: RUTUJA KAUTHALE, ROLL: 221058, BATCH: A3, GR NO. : 17U251