

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

ASSIGNMENT 1

AIM: To Create ADT that implement the "set" concept.

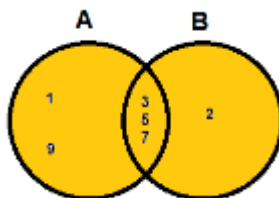
OBJECTIVE: To Create ADT that implement the "set" concept a] Add b] Remove c] Contains d] Size e] Intersection of two set f] Union of two sets g] Difference between two sets h] subset .

THEORY:

Union

The union of two sets is a new set that contains all of the elements that are in at least one of the two sets.

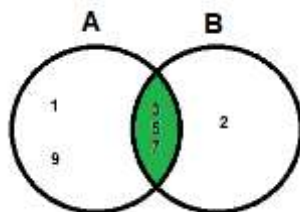
The union is written as $A \cup B$ or "A or B or B".



Intersection

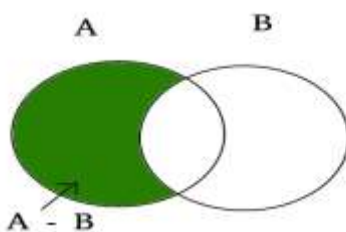
The intersection of two sets is a new set that contains all of the elements that are in both sets. The

intersection is written as $A \cap B$ or "A and B or B".



DIFFERENCE

Set difference is a generalization of the idea of the complement of a set and as such is sometimes called the relative complement of T with respect to S. The symmetric difference between two sets S and T is the union of $S - T$ and $T - S$. "set difference." A Dictionary of Computing. . "set difference."



SUBSET

A subset is a set whose elements are all members of another set.

The symbol " \subseteq " means "is a subset of".

The symbol " \subset " means "is a proper subset of".

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

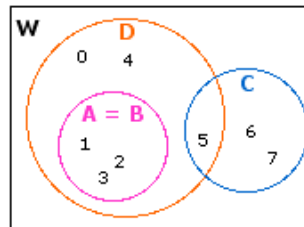
W: The whole numbers: {0, 1, 2, 3, ...}

A: {1, 2, 3}

B: {1, 2, 3}

C: {5, 6, 7}

D: {0, 1, 2, 3, 4, 5}



PROGRAM:

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

class Set
{
    int set[20];
    int size;
public:
    Set(int s)
    {
        size=s;
    }

    void create()
    {
        for(int i=0;i<size;i++)
        {
            cin>>set[i];
        }
    }

    int insert(int element)
    {
        set[size]=element;
        return ++size;
    }
}
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
    }

    int contains(int element)
    {
        int i=0;
        while(i<size)
        {
            if(set[i]==element)
                return i+1;
            i++;
        }
        return 0;
    }

    int set_size()
    {
        return size;
    }

    void remove(int element)
    {
        int i=0;
        while(i<size)
        {
            if(set[i]==element)
            {
                for(int j=i;j<size;j++)
                {
                    set[j]=set[j+1];
                }
                cout<<"\nElement deleted...";
                size=size-1;
                return;
            }
        }
    }
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        i++;
    }
    cout<<"\nSorry, element not found...";
}
```

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

```
void set_intesection(Set &set1, Set &set2)
{
    int i=0, j=0;
    while(i<set2.size)
    {
        if(compare(set2.set[i], set1))
        {
            this->set[j]=set2.set[i];
            j++;
        }
        i++;
    }
    size=j;
}
```

```
void set_union(Set &set1, Set &set2)
{
    int j;
    for(j=0; j<set1.size; j++)
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        set[j]=set1.set[j];

    int i=0;
    while(i<set2.size)
    {
        if(!compare(set2.set[i],set1))
        {
            this->set[j]=set2.set[i];
            j++;
        }
        i++;
    }
    size=j;
}

void set_diff(Set &set1,Set &set2)
{
    int i=0,j=0;
    while(i<set1.size)
    {
        if(!compare(set1.set[i],set2))
        {
            this->set[j]=set1.set[i];
            j++;
        }
        i++;
    }
    size=j;
}

int subset(Set &s)
{
    int i=0,status=1;
    while(i<s.size)
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        {
            if(!compare(s.set[i],*this))
            {
                status=0;
                break;
            }
            else
                status=1;

            i++;
        }
        return status;
    }

    int compare(int e,Set &s)
    {
        for(int i=0;i<s.size;i++)
        {
            if(e==s.set[i])
                return 1;
        }
        return 0;
    }

};

int main()
{
    Set inter(20),BmA(20),AmB(20),un(30);
    int size1,size2,ch,status,e;
    char set_ch;
    char tryAgain;
    cout<<"Enter the size of set A: ";
    cin>>size1;
    Set s1(size1);
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
cout<<"Enter the elements of set A: ";
s1.create();
cout<<"Enter the size of set B: ";
cin>>size2;
Set s2(size2);
cout<<"Enter the elements of set B: ";
s2.create();
cout<<"Set A: ";
s1.display();
cout<<"\nSet B: ";
s2.display();
do{
    cout<<"\n|=====Menu=====|";
    cout<<"\n1.Insert";
    cout<<"\n2.Remove";
    cout<<"\n3.Contains";
    cout<<"\n4.Size";
    cout<<"\n5.Intersection";
    cout<<"\n6.Union";
    cout<<"\n7.A-B";
    cout<<"\n8.B-A";
    cout<<"\n9.Check subset";
    cout<<"\n10.Display Sets.";
    cout<<"\n11.Exit.";
    cout<<"\nEnter your choice: ";
    cin>>ch;

    switch(ch)
    {
    case 1:
        cout<<"Which set you want to enter the element in?(A/B): ";
        cin>>set_ch;
        cout<<"Enter element to be inserted: ";
        cin>>e;
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        if(set_ch=='A' || set_ch=='a')
        {
            s1.insert(e);
            cout<<"\nSet A: ";
            s1.display();
        }
        else
        {
            s2.insert(e);
            cout<<"\nSet B: ";
            s2.display();
        }
    }
    break;

case 2:
    cout<<"Which set you want to delete the element from?(A/B): ";
    cin>>set_ch;
    cout<<"Enter element to be deleted: ";
    cin>>e;
    if(set_ch=='A' || set_ch=='a')
    {
        s1.remove(e);
        cout<<"\nSet A: ";
        s1.display();
    }
    else
    {
        s2.remove(e);
        cout<<"\nSet B: ";
        s2.display();
    }
    break;

case 3:
```


NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
int pos;
cout<<"In which set you want check?(A/B): ";
cin>>set_ch;
cout<<"Enter element to be checked: ";
cin>>e;
if(set_ch=='A' || set_ch=='a')
{
    pos=s1.contains(e);
    if(pos!=0)
    {
        cout<<"\nElement found at position: "<<pos;
        cout<<"\nSet A: ";
        s1.display();
        break;
    }
}
else
{
    pos=s2.contains(e);
    if(pos!=0)
    {
        cout<<"\nElement found at position: "<<pos;
        cout<<"\nSet B: ";
        s2.display();
        break;
    }
}
cout<<"\nSorry,element not found...";
break;

case 4:
    cout<<"\nSet A: ";
    s1.display();
    size1=s1.set_size();
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        cout<<" with Size: "<<size1;
        cout<<"\nSet B: ";
        s2.display();
        size2=s2.set_size();
        cout<<" with Size: "<<size2;
    break;

    case 5:
        inter.set_intesection(s1,s2);
        cout<<"\nIntersection: ";
        inter.display();
    break;

    case 6:
        un.set_union(s1,s2);
        cout<<"\nUnion: ";
        un.display();
    break;

    case 7:
        AmB.set_diff(s1,s2);
        cout<<"\nA-B: ";
        AmB.display();
    break;

    case 8:
        BmA.set_diff(s2,s1);
        cout<<"\nB-A: ";
        BmA.display();
    break;

    case 9:
        status=s1.subset(s2);
        (status==0)?cout<<"\nB Subset of A: false":cout<<"\nB Subset of A: true";
```

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

```

        break;
    case 10:
        cout<<"Set A: ";
        s1.display();
        cout<<"\nSet B: ";
        s2.display();

        break;
    case 11:
        exit(1);

    default: cout<<"\nInvalid Entry...";
}

cout<<"\n\nDo you want to try again? (Y/N)";
cin>>tryAgain;
}while(tryAgain=='Y' || tryAgain=='y');
}

```

OUTPUT:

```

Enter the size of set A: 5
Enter the elements of set A: 1 2 3 4 5
Enter the size of set B: 4
Enter the elements of set B: 2 5 6 8
Set A: { 1 2 3 4 5 }
Set B: { 2 5 6 8 }
[.....]
1. Insert
2. Remove
3. Contains
4. Size
5. Intersection
6. Union
7. A-B
8. B-A
9. Check subset
10. Display Sets.
11. Exit.
Enter your choice: 5
Intersection: { 2 5 }

Do you want to try again? (Y/N) y
[.....]
1. Insert
2. Remove
3. Contains
4. Size
5. Intersection
6. Union
7. A-B
8. B-A
9. Check subset
10. Display Sets.
11. Exit.
Enter your choice: 6
Union: { 1 2 3 4 5 6 8 }

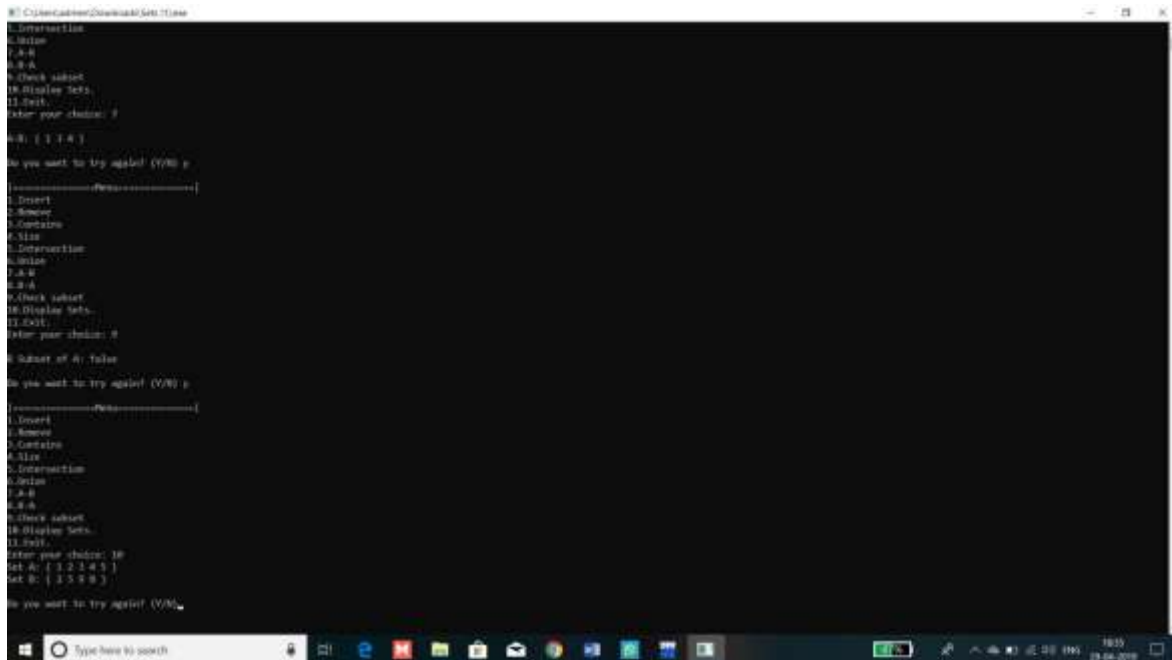
Do you want to try again? (Y/N) y
[.....]
1. Insert
2. Remove
3. Contains
4. Size
5. Intersection
6. Union
7. A-B

```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062



CONCLUSION:

Hence, we implemented set identity laws in C++ and learnt about its properties.

ASSIGNMENT 2

AIM: Construct a threaded binary search tree.

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

THEORY: Insertion in Binary threaded tree is similar to insertion in binary tree but we will have to adjust the threads after insertion of each element. The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion. A binary tree is made threaded by making all right child pointers that would normally be NULL point to the inorder successor of the node (if it exists).

ALGORITHM:

Insertion Operation

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

Step 1: Start

Step 2: If the root is null then

create root node

return

Step 3: If the root exists then

Compare the data with root.data

Step 4: While insert position is located

If data is greater than node.data

Go to right subtree

Else

Go to left subtree

End while

Step 5: Insert data

Step 6: Stop.

Inorder Traversal

Step 1: Start

Step 2: current=leftmost->root

Step 3: While current is not null OR current is not equal to head

Print current->data.

Step 4: If current->rbit is true then

current= current->right

else current=leftmost->(current-right)

Step 5: Stop.

PROGRAM:

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

```
#include<iostream>

using namespace std;

struct Node
{
    struct Node *left, *right;
    int info;
    int rf,lf=0;
};

struct Node *insert(struct Node *root, int key)
{
    Node *temp = root;
    Node *par = NULL;
    while (temp!= NULL)
    {
        if (key == (temp->info))
        {
            cout<<"Duplicate Key !\n";
            return root;
        }
        par = temp;
        if (key < temp->info)
        {
            if (temp -> lf == 0)
            temp = temp -> left;
            else
            break;
        }
    }
}
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
}  
  
else  
  
{  
  
if (temp->rf == 0)  
temp = temp -> right;  
  
else  
  
break;  
  
}  
  
}  
  
Node *tmp = new Node;  
  
tmp -> info = key;  
  
tmp -> lf = 1;  
  
tmp -> rf = 1;  
  
if (par == NULL)  
{  
  
root = tmp;  
  
tmp -> left = NULL;  
  
tmp -> right = NULL;  
  
}  
  
else if (key < (par -> info))  
{  
  
tmp -> left = par -> left;  
  
tmp -> right = par;  
  
par -> lf = 0;  
  
par -> left = tmp;  
  
}
```

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

```
else
{
tmp -> left = par;
tmp -> right = par -> right;
par -> rf = 0;
par -> right = tmp;
}

return root;
}

struct Node *inorderSuccessor(struct Node *ptr)
{
if (ptr -> rf == 1)
return ptr->right;
ptr = ptr -> right;
while (ptr -> lf == 0)
ptr = ptr -> left;
return ptr;
}

void inorder(struct Node *root)
{
if (root == NULL)
cout<<"Tree is empty";
struct Node *ptr = root;
while (ptr -> lf == 0)
ptr = ptr -> left;
```


NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

```
while (ptr != NULL)
{
cout<<ptr->info<<"\t";
ptr = inorderSuccessor(ptr);
}
}

int main()
{
struct Node *root = NULL;
root = insert(root, 20);
root = insert(root, 10);
root = insert(root, 30);
root = insert(root, 5);
root = insert(root, 16);
root = insert(root, 14);
root = insert(root, 17);
root = insert(root, 13);
inorder(root);
return 0;
}
```

OUTPUT:

5 10 13 14 16 17 20 30

CONCLUSION: Successfully implemented a TBT, inserted values in given order and traversed it in inorder traversal using threads.

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

Assignment No: 3

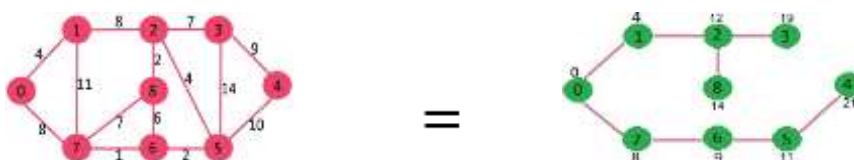
Aim : There are flight paths between cities. If there is a flight between city A and city B then there is an edge between the cities. The cost of the edge can be the time that flight takes to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node can be represented 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.

Objectives: To understand the various operations on graphs.

Theory:

Dijkstra's algorithm is very similar to Prim's algorithm for minimum spanning tree. Like Prim's MST, we generate a shortest path tree with given source as root. We maintain two sets, one set contains vertices included in shortest path tree, and other set includes vertices not yet included in shortest path tree.

For Example:



At every step of the algorithm, we find a vertex which is in the other set and has a minimum distance from the source. Below are the detailed steps used in Dijkstra's algorithm to find the shortest path from a single source vertex to all other vertices in the given graph.

Algorithm:

1. Create priority queue pq
2. Enqueue(pq,s)
3. For(i=1;i<=g->v;i++)

Distance[i]=-1
4. Distance[s]=0

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
5. while(!isemptyqueue(pq))
{
    5.1 v=deletemin(pq);
    5.2 for all adjacent vertices w to v
    {
        Compute new distance d=distance[v]+weight[v][w];
        If(Distance[w]==-1)
        {
            Distance[w]=new distance d;
            Insert w in priorityqueue with priority d
            Path[w]=v
        }
        If(Distance[w]>newdistance d)
        {
            distance[w]=new distance d;
            Update priority Of vertex w to be d;
            Path[w]=v;
        }
    }
}
```

Program:

```
#include<iostream>
#define MAX 20
using namespace std;

class dijkstra
{
    int city;
    int distance[MAX][MAX];
    int d[MAX];
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        int visited[MAX];
public:
    void city_no();
    int minvertex();
    void matrix_fill();
    void dijkstra_code();
    void display();
};

void dijkstra::city_no()
{
    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()
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
{
    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:

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
E:\Sem 4 Practical Assignments\SD\sd3.exe
enter the number of cities (including cities A and B) : 5
enter the distances between the cities :
For city 0
0 4 8 0 0
For city 1
4 0 2 5 0
For city 2
8 2 0 5 9
For city 3
0 5 5 0 4
For city 4
0 0 9 4 0
distance of cities from city 0
city Distance
0      0
1      4
2      0
3      9
4     13
*****
Process exited after 63.72 seconds with return value 0
Press any key to continue . . .
```

Conclusion: From above experiment we learnt how to use shortest path algorithm using graph operation.

ASSIGNMENT 4

AIM: To implement prims algorithm for minimum spanning tree.

OBJECTIVE: For a weighted graph G, find the minimum spanning tree using prims algorithm.

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

THEORY:

In computer science, Prim's (also known as Jarník's) algorithm is a greedy algorithm that finds a minimum spanning tree for a weighted undirected graph. This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized. The algorithm operates by building this tree one vertex at a time, from an arbitrary starting vertex, at each step adding the cheapest possible connection from the tree to another vertex.

The algorithm was developed in 1930 by Czech mathematician Vojtěch Jarník and later rediscovered and republished by computer scientists Robert C. Prim in 1957 and Edsger W. Dijkstra in 1959. Therefore, it is also sometimes called the Jarník's algorithm, Prim–Jarník algorithm, Prim–Dijkstra algorithm or the DJP algorithm.

ALGORITHM:

- 1) Create a set *mstSet* that keeps track of vertices already included in MST.
- 2) Assign a key value to all vertices in the input graph. Initialize all key values as INFINITE. Assign key value as 0 for the first vertex so that it is picked first.
- 3) While *mstSet* doesn't include all vertices
 -a) Pick a vertex *u* which is not there in *mstSet* and has minimum key value.
 -b) Include *u* to *mstSet*.
 -c) Update key value of all adjacent vertices of *u*. To update the key values, iterate through all adjacent vertices. For every adjacent vertex *v*, if weight of edge *u-v* is less than the previous key value of *v*, update the key value as weight of *u-v*

The idea of using key values is to pick the minimum weight edge from cut. The key values are used only for vertices which are not yet included in MST, the key value for these vertices indicate the minimum weight edges connecting them to the set of vertices included in MST.

PROGRAM:

```
#include <iostream>

using namespace std;

class graph
{
    int a[100][100];

    int v;

public:

    void insert_edge(int n1,int n2,int wt)
    {
        if(n1-1>=v | n2-1>=v)
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        cout<<"Vertex request out of range\n";

    else

    {

        a[n1-1][n2-1]=wt;

        a[n2-1][n1-1]=wt;

    }

}

void display()

{

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

    {

        for(int j=0;j<v;j++)

        {

            cout<<a[i][j]<<"\t";

        }

        cout<<endl;

    }

}

void update_v(int n)

{

    v=n;

}

void prims(int src)
```


NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
{  
  
    int sp[v],dist[v],visited[v],parent[v],c=0;  
  
    for(int i=0;i<v;i++)  
  
    {  
  
        visited[i]=0;  
  
        dist[i]=9999;  
  
    }  
  
    dist[src-1]=0;  
  
    parent[src-1]=-1;  
  
    for(int i=0;i<v;i++)  
  
    {  
  
        int min=9999,min_ind;  
  
        for(int j=0;j<v;j++)  
  
        {  
  
            if(!visited[j] && dist[j]<min )  
  
            {  
  
                min=dist[j];  
  
                min_ind=j;  
  
            }  
  
        }  
  
        int U=min_ind;  
  
        visited[U]=1;
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        sp[c]=U;

        c++;

        for(int V=0;V<v;V++)

        {

            if(!visited[V] && a[U][V] && a[U][V]<dist[V] && dist[U]!=9999)

            {

                parent[V]=U;

                dist[V]=a[U][V];

            }

        }

    }

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

    {

        cout<<sp[i]+1<<" link from "<<parent[i]+1<<endl;

    }

    cout<<endl;

}

};

int main()

{

    char r;

    do

    {
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
graph g;

char op;

int v;

cout<<"Enter number of vertices: ";

cin>>v;

g.update_v(v);

do

{

    int c;

    cout<<"\n=====Menu=====\\n";

    cout<<"1] Insert edge\\n2] Increase number of vertices\\n3] Display matrix\\n4] Find shortest path\\n";

    cout<<"_____\\n";

    cout<<"Enter your choice: ";

    cin>>c;

    switch(c)

    {

        case 1: {

            int n1,n2,wt;

            cout<<"Enter the nodes between which there is an edge\\n";

            cin>>n1>>n2;

            cout<<"Enter weight: ";

            cin>>wt;

            g.insert_edge(n1,n2,wt);
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        }

        break;

    case 2: {

        int n;

        cout<<"Enter the number by which you wish to increase the vertices: ";

        cin>>n;

        v+=n;

        g.update_v(v);

    }

    break;

    case 3: {

        g.display();

    }

    break;

    case 4: {

        int src,dst;

        cout<<"Source: ";

        cin>>src;

        g.prims(src);

    }

    break;

    default:cout<<"Error 404.....page not found\n";

}
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
    cout<<"Do you wish to continue(y/n): ";

    cin>>op;

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

    cout<<"Test pass(y/n): ";

    cin>>r;

    }while(r=='n' || r=='N');

    cout<<"*****\n";

    cout<<"*   Thank You!   *\n";

    cout<<"*****\n";

    return 0;

}
```

OUTPUT:

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062



```
C:\Users\ashish\Documents\PROGRAM4A_primew [Swedish] - Dev-C++ 5.11
Do you wish to continue(y/n): y

-----Menu-----
1) Insert edge
2) Increase number of vertices
3) Display matrix
4) Find shortest path

Enter your choice: 3
0 4 5 0
0 0 0 0
0 0 0 0
0 0 0 0
Do you wish to continue(y/n): y

-----Menu-----
1) Insert edge
2) Increase number of vertices
3) Display matrix
4) Find shortest path

Enter your choice: 4
Source: 0
1 link from 0 to 0
2 link from 1
3 link from 1
4 link from 1
Do you wish to continue(y/n):
```

CONCLUSION:

Time Complexity of the above program is $O(V^2)$. If the input graph is represented using adjacency list, then the time complexity of Prim's algorithm can be reduced to $O(E \log V)$ with the help of binary heap.

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

ASSIGNMENT 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.

Objective: Understand the problem statement, determine and implement the data structure suitable for solving above real time example.

Theory:

Kruskal's Minimum Spanning Tree:

Kruskal's algorithm is a minimum-spanning-tree **algorithm** which finds an edge of the least possible weight that connects any two trees in the forest. ... This means it finds a subset of the edges that forms a tree that includes every vertex, where the total weight of all the edges in the tree is minimized.

Algorithm:

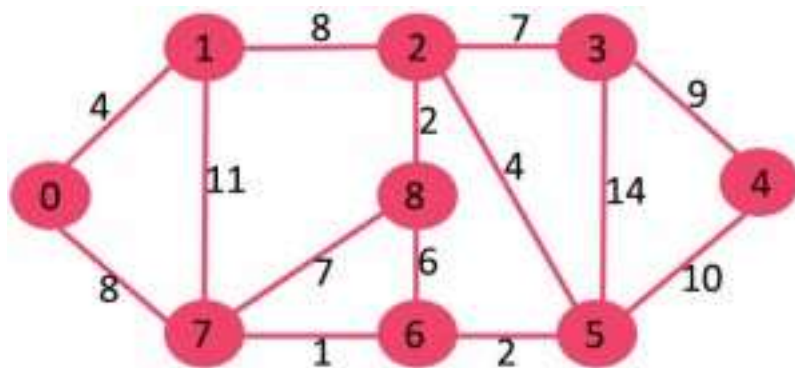
- KRUSKAL(G):
- $A = \emptyset$ For each vertex $v \in G.V$:
- MAKE-SET(v)
- For each edge $(u, v) \in G.E$ ordered by increasing order by weight(u, v):
- if FIND-SET(u) \neq FIND-SET(v):
- $A = A \cup \{(u, v)\}$
- UNION(u, v)
- return A

Example:

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062



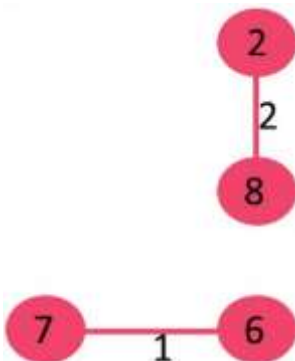
The graph contains 9 vertices and 14 edges. So, the minimum spanning tree formed will be having $(9 - 1) = 8$ edges.

Now pick all edges one by one from sorted list of edges

1. Pick edge 7-6: No cycle is formed, include it.



2. Pick edge 8-2: No cycle is formed, include it.

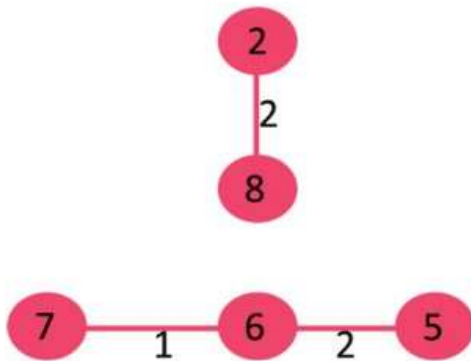


3. Pick edge 6-5: No cycle is formed, include it.

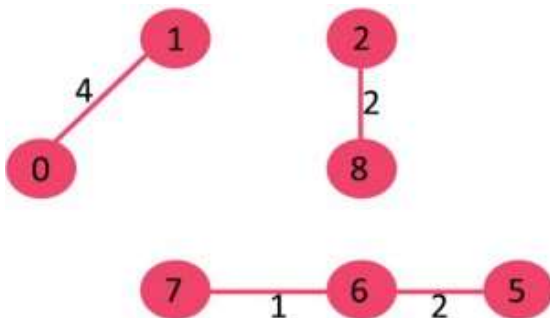
NAME: Ashish Khurkhuriya

Gr.No: u1610120

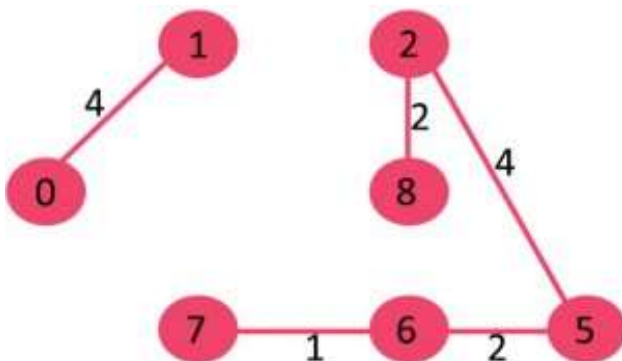
Roll. No: 223062



4. Pick edge 0-1: No cycle is formed, include it.



5. Pick edge 2-5: No cycle is formed, include it.



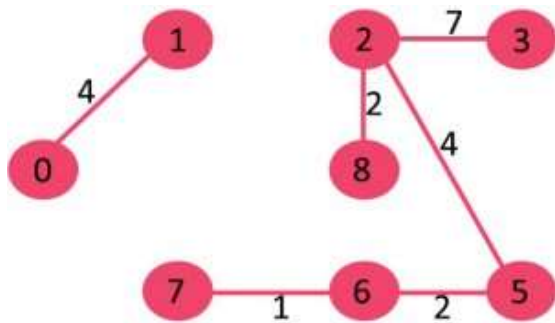
NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

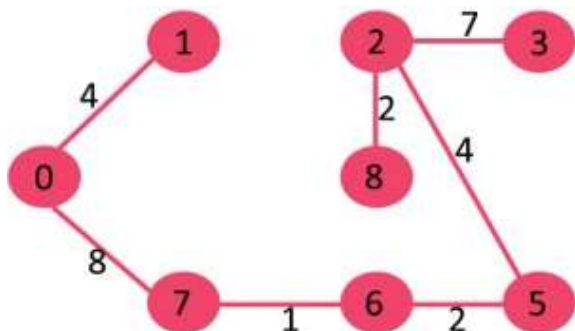
6. Pick edge 8-6: Since including this edge results in cycle, discard it.

7. Pick edge 2-3: No cycle is formed, include it.



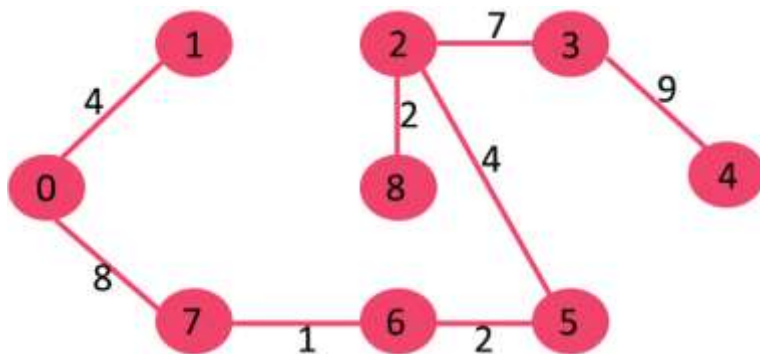
8. Pick edge 7-8: Since including this edge results in cycle, discard it.

9. Pick edge 0-7: No cycle is formed, include it.



10. Pick edge 1-2: Since including this edge results in cycle, discard it.

11. Pick edge 3-4: No cycle is formed, include it.



Since the number of edges included equals $(V - 1)$, the algorithm stops here.

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

Program:

```
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();
```

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

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

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

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;
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
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);
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        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;
}
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
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;
    }
}
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
}

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

}
```

Output:

Enter number of city's:6

Enter the adjacency matrix of city ID's:

0 3 1 6 0 0

3 0 5 0 3 0

1 5 0 5 6 4

6 0 5 0 0 2

0 3 6 0 0 6

0 0 4 2 6 0

2 0 1

5 3 2

1 0 3

4 1 3

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

5 2 4

Minimum cost of the telephone lines between the cities: 13

CONCLUSION:

Thus, we learnt about **Kruskal's algorithm** is a minimum-spanning-tree **algorithm** which finds an edge of the least possible weight that connects any two trees in the forest. And its application for traversing each node with the shortest distance to reach it.

NAME: Ashish Khurkhuriya
Gr.No: u1610120
Roll. No: 223062

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.

Objective:

Understand the heap data structure for heap sorting.

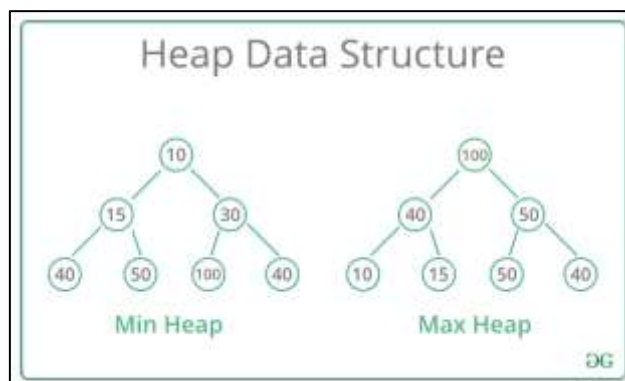
Theory:

Heap:

A Heap is a special Tree-based data structure in which the tree is a complete binary tree. Generally, Heaps can be of two types:

Max-Heap: In a Max-Heap the key present at the root node must be greatest among the keys present at all of its children. The same property must be recursively true for all sub-trees in that Binary Tree.

Min-Heap: In a Min-Heap the key present at the root node must be minimum among the keys present at all of its children. The same property must be recursively true for all sub-trees in that Binary Tree.

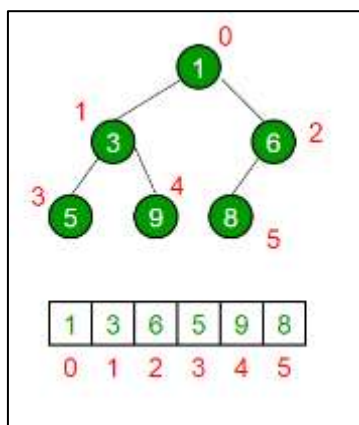


NAME: Ashish Khurkhuriya

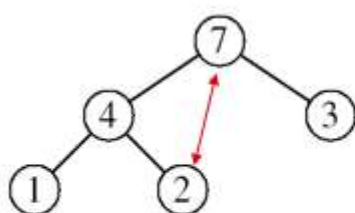
Gr.No: u1610120

Roll. No: 223062

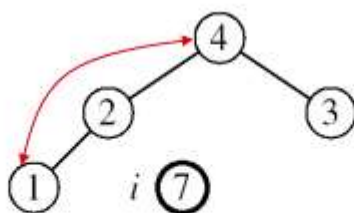
Since a Binary Heap is a Complete Binary Tree, it can be easily represented as array and array based representation is space efficient. If the parent node is stored at index I , the left child can be calculated by $2 * I + 1$ and right child by $2 * I + 2$ (assuming the indexing starts at 0).



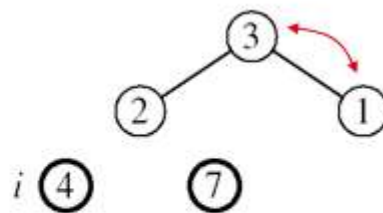
EXAMPLE: $A = [7, 4, 3, 1, 2]$



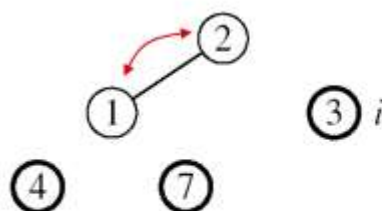
MAX-HEAPIFY(A, 1, 4)



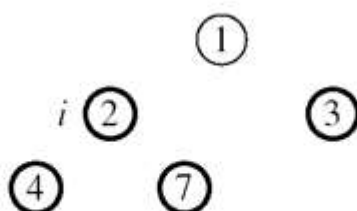
MAX-HEAPIFY(A, 1, 3)



MAX-HEAPIFY(A, 1, 2)



MAX-HEAPIFY(A, 1, 1)



A [1 | 2 | 3 | 4 | 7]

Algorithm:

1. MAX-HEAPIFY(A, i, n)

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
l ← LEFT(i)
r ← RIGHT(i)
if l ≤ n and A[l] > A[i]
    then largest ← l
    else largest ← i
if r ≤ n and A[r] > A[largest]
    then largest ← r
if largest != i
    then exchange A[i] ↔ A[largest]
    MAX-HEAPIFY(A, largest, n)
```

2. BUILD-MAX-HEAP(A)

```
n = length[A]
for i ← floor(n/2) downto 1
    do MAX-HEAPIFY(A, i, n)
```

3. HEAPSORT(A)

```
BUILD-MAX-HEAP(A)
for i ← length[A] downto 2
    do exchange A[1] ↔ A[i]
    MAX-HEAPIFY(A, 1, i - 1)
```

Code:

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

void max_heapify(int *arr, int i, int n) {
    int left_num = (2*i);
    int right_num = (2*i) + 1;
    int largest = i;

    if(left_num <= n && arr[left_num] > arr[i]) {
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        largest = left_num;
    } else {
        largest = i;
    }
    if(right_num <= n && arr[right_num] > arr[largest]) {
        largest = right_num;
    }
    if(largest != i) {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        max_heapify(arr, largest, n);
    }
}

void build_heap(int *arr, int n) {
    for(int i=floor(n/2); i>=0; i--) {
        max_heapify(arr,i,n);
    }
}

int main() {
    int n;
    cout << "Enter no. of students : ";
    cin >> n;
    int *arr = new int[n];
    cout << "Enter Marks of M3 : \n";
    for(int i=1;i<=n;i++) {
        cin >> arr[i];
    }
    //HEAP SORT
    build_heap(arr,n);
    for(int i=n;i>=1;i--) {
        int temp = arr[1];
        arr[1] = arr[i];
        arr[i] = temp;

        max_heapify(arr,1,i-1);
    }
    cout << "Maximum Marks Obtained = " << arr[n] << endl;
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        cout << "Minimum Marks Obtained = " << arr[1] << endl;  
        return 0;  
    }
```

Output:

```
Enter no. of students : 6  
Enter Marks of M3 :  
20  
28  
29  
17  
15  
22  
Maximum Marks Obtained = 29  
Minimum Marks Obtained = 15
```

Conclusion:

In this assignment, we have learnt the usage of heap data structure for heap sorting.

ASSIGNMENT NO 7

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

OBJECTIVE: To understand the concept of double hashing and insert the keys into hash table.

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

THEORY: Double hashing is a [computer programming](#) technique used in [hash tables](#) to resolve [hash collisions](#), in cases when two different values to be searched for produce the same hash key. It is a popular [collision](#)-resolution technique in [open-addressed](#) hash tables. Double hashing is implemented in many popular [libraries](#).

Like [linear probing](#), it uses one hash value as a starting point and then repeatedly steps forward an interval until the desired value is located, an empty location is reached, or the entire table has been searched; but this interval is decided using a second, independent hash function (hence the name double hashing). Unlike [linear probing](#) and [quadratic probing](#), the interval depends on the data, so that even values mapping to the same location have different bucket sequences; this minimizes repeated collisions and the effects of clustering.

Given two randomly, uniformly, and independently selected hash functions h_1 and h_2 , the i th location in the bucket sequence for value k in a hash table T is: $h(i,k) = (h_1(k) + i \cdot h_2(k)) \bmod [T]$. Generally, h_1 and h_2 are selected from a set of universal hash functions

Double Hashing Example:

Insert 89, 18, 49, 58, 69 in the table

$$\text{Hash1}(\text{key}) = \text{key} \bmod 10$$

$$\text{Hash2}(\text{key}) = 7 - (\text{key} \bmod 7)$$

$$H(89) = 89 \bmod 10 = 9$$

$$H(18) = 18 \bmod 10 = 8$$

$$\begin{aligned} H(49) &= 49 \bmod 10 = 9 \rightarrow \text{collision} \\ &= 7 - (49 \bmod 7) = 7 \end{aligned}$$

$$\begin{aligned} H(58) &= 58 \bmod 10 = 8 \rightarrow \text{collision} \\ &= 7 - (58 \bmod 7) = 5 \end{aligned}$$

$$\begin{aligned} H(69) &= 69 \bmod 10 = 9 \rightarrow \text{collision} \\ &= 7 - (69 \bmod 7) = 1 \end{aligned}$$

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

[0]	
[1]	69
[2]	
[3]	
[4]	
[5]	58
[6]	
[7]	49
[8]	18
[9]	89

ALGORITHM:

1. Set $\text{indx} = H(K)$; $\text{offset} = H_2(K)$
2. If table location indx already contains the key, no need to insert it. Done!
3. Else if table location indx is empty, insert key there. Done!
4. Else collision. Set $\text{indx} = (\text{indx} + \text{offset}) \bmod M$.
5. If $\text{indx} == H(K)$, table is full! (Throw an exception, or enlarge table.) Else go to 2.

CODE:

```
#include <iostream>

using namespace std;

int hashing(int a[],int value,int n)
{
    int i=1;

    int key = value%n;

    if (a[key]==-1)
        a[key]=value;

    else
    {
        while(a[key]!=-1)
```


NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
{
    key=i*(1+value%(n-1));
    i+=1;
}

if (a[key]==-1 )
a[key]=value;
else
cout<<" no index to store value "<<endl;
}

return 0;
}

void print(int a[],int n)
{
    cout<<"Index "<<"      "<<" Value "<<endl;
    for (int i=0;i<n;i++)
    {
        if (a[i]!=-1)
            cout<<i<<"      "<<a[i]<<endl;
    }
}
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
int sizeofset(int a[],int n)
{
    int cnt=0;
    for(int i=0;i<n;i++)
    {
        if (a[i]!=-1)
            cnt+=1;
    }
    return cnt;
}

int searchs(int a[],int svalue,int n)
{
    int i=1;
    int key=svalue%n;
    if (a[key]==svalue)
        return key;
    else
    {
        while(a[key]!=svalue)
        {
            key=i*(1+svalue%(n-1));
            i+=1;
        }
        if (a[key]== svalue)
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        return key;

    else

        return -1;

    }

}

int main()

{ int value,s,n;

    char choice='y';

    cout<<" enter the size of hashtable "<<endl;

        cin>>n;

        int a[n];

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

            a[i]=-1;

    while (choice=='y')

    {

        cout<<endl<<"..... MENU ..... "<<endl;

        cout<<" enter 1 for read number of element and factor present in hashtable "<<endl;

        cout<<" enter 2 for to add value into the hashtable "<<endl;

        cout<<" enter 3 to print the hashtable "<<endl;

        cout<<" enter 4 to search the element "<<endl;

        cout<<" enter 5 to exit "<<endl;

        cin>>s;

        switch(s)
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
{
```

case 1:

```
cout<<" total element in the set are "<<sizeofset(a,n)<<endl;
```

```
cout<<" Load factor of hashtable is "<<sizeofset(a,n)/n<<endl;
```

```
break;
```

case 2:

```
cout<<endl<<" enter the value to store in hashtable "<<endl;
```

```
cin>>value;
```

```
hashing(a,value,n);
```

```
break;
```

case 3 :

```
print(a,n);
```

```
break;
```

case 4:

```
int svalue;
```

```
cout<<" enter the value to search "<<endl;
```

```
cin>>svalue;
```

```
if (searchs(a,svalue,n)!=-1)
```

```
cout<<" value "<<svalue<<" is present at the key "<<searchs(a,svalue,n)<<endl;
```

```
else
```

```
cout<<" element not present in the hashtable "<<endl;
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        break;

case 5:
    choice= 'n';
    break;
}
}

cout << "Hello world!" << endl;

return 0;
}
```

OUTPUT:

```
/*/
```

enter the size of hashtable

100

..... MENU

enter 1 for read number of element and factor present in hashtable

enter 2 for to add value into the hashtable

enter 3 to print the hashtable

enter 4 to search the element

enter 5 to exit

2

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

enter the value to store in hashtable

26

..... MENU

enter 1 for read number of element and factor present in hashtable

enter 2 for to add value into the hashtable

enter 3 to print the hashtable

enter 4 to search the element

enter 5 to exit

2

enter the value to store in hashtable

36

..... MENU

enter 1 for read number of element and factor present in hashtable

enter 2 for to add value into the hashtable

enter 3 to print the hashtable

enter 4 to search the element

enter 5 to exit

2

enter the value to store in hashtable

126

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

..... MENU

enter 1 for read number of element and factor present in hashtable

enter 2 for to add value into the hashtable

enter 3 to print the hashtable

enter 4 to search the element

enter 5 to exit

2

enter the value to store in hashtable

226

..... MENU

enter 1 for read number of element and factor present in hashtable

enter 2 for to add value into the hashtable

enter 3 to print the hashtable

enter 4 to search the element

enter 5 to exit

2

enter the value to store in hashtable

136

..... MENU

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

enter 1 for read number of element and factor present in hashtable

enter 2 for to add value into the hashtable

enter 3 to print the hashtable

enter 4 to search the element

enter 5 to exit

3

Index	Value
-------	-------

26	26
----	----

28	126
----	-----

29	226
----	-----

36	36
----	----

38	136
----	-----

..... MENU

enter 1 for read number of element and factor present in hashtable

enter 2 for to add value into the hashtable

enter 3 to print the hashtable

enter 4 to search the element

enter 5 to exit

4

enter the value to search

226

value 226 is present at the key 29

/*/

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

Assignment 8

Aim:

Store information of student in sequential file. It should contain roll no, name, division address. Allow user to add, delete, search information of student.

Objective:

Understand sequential file organization, implement it using C++.

Theory: The simplest organization for a file is sequential. A *sequential file* is a sequence of records. The records may or may not be kept in sorted order in the sequence. In standard C input/output all files are sequential files. A record of a file is not necessarily declared to be of type structure. Although file records are typically of type structure, a file record may also be declared to be of type integer, float, character, or any other C type. All records of a file need not be the same type.

Advantages of sequential file

- It is simple to program and easy to design.
- Sequential file is best use if storage space.

Disadvantages of sequential file

- Sequential file is time consuming process.
- It has high data redundancy.
- Random searching is not possible.

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

Program:

```
#include <iostream>

#include <fstream>

#include <string>

using namespace std;

int main()

{   int k=0;

string line;

string find;

char name[25];

int roll=0;

char d;

string adres;

ofstream myfile,f2;

myfile.open("data.txt",ios::app);

while(k!=4){

cout<<"press 1 for adding data"<<endl;

cout<<"press 2 for update "<<endl;

cout<<"press 3 for searching"<<endl;

cin>>k;

if(k==1)

{

cout<<"enter roll no. "<<endl;
```

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

```
cin>>roll;

cout<<"enter Name"<<endl;

    cin>>name;

cout<<"enter division "<<endl;

cin>>d; cout<<"enter adre."<<endl;

cin>>adres; myfile<<roll<<" ";

myfile<<name<<" ";

myfile<<d<<" ";

myfile<<adres<<endl;

}

if(k==2)

{

cout<<"Enter roll no, you want to update "<<endl;

cin>>find;

ifstream file,f1;

file.open("data.txt",ios::in);

file>>line;

while (!file.eof() && line!=find)

{

getline(file,line);

}

if(find!=line)
```

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

```
{  
  
cout<<"deatAILS NOT FOUND";  
  
}  
  
else{  
  
cout<<"enter Name"<<endl;  
  
cin>>name;  
  
cout<<"enter dividion "<<endl;  
  
cin>>d;  
  
cout<<"enter address"<<endl;  
  
cin>>adres; myfile<<find;  
  
myfile<<name;  
  
myfile<<d;  
  
myfile<<adres<<endl;  
  
}}  
  
if(k==3)  
  
{  
  
cout<<"Enter roll no, you want to update "<<endl;  
cin>>find;  
  
ifstream file;  
  
file.open("data.txt",ios::in);  
  
file>>line;  
  
while (!file.eof() && line!=find)  
  
{
```

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062

```
getline(file,line); file>>line;

}

if(find!=line)

{

cout<<"deatAILS NOT FOUND";

}

else {

cout<<"roll.no"<<endl;

cout<<line<<endl;

file>>line;

cout<<line<<endl;

file>>line;

cout<<line<<endl;

file>>line;

cout<<line<<endl; }

}

}

return 0; }
```

Output:

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
krishna@krishna-K30AD-R31AD-M31AD:~$ g++ sfile.cpp
krishna@krishna-K30AD-R31AD-M31AD:~$ ./a.out
press 1 for adding data
press 2 for update
press 3 for searching
3
[20] Stopped ./a.out
krishna@krishna-K30AD-R31AD-M31AD:~$ g++ sfile.cpp
krishna@krishna-K30AD-R31AD-M31AD:~$ ./a.out
press 1 for adding data
press 2 for update
press 3 for searching
3
Enter roll no, you want to update
1
roll.no
1
shivam
laxminagar, pune.
press 1 for adding data
press 2 for update
press 3 for searching
3
Enter roll no, you want to update
2
roll.no
2
nana
balajinagar, pune.
press 1 for adding data
press 2 for update
press 3 for searching
4
krishna@krishna-K30AD-R31AD-M31AD:~$ ./a.out
press 1 for adding data
press 2 for update
press 3 for searching
```

Conclusion: In this assignment, we have learnt the sequential file organization.

ASSIGNMENT 9

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

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:

64-bit Open source Linux or its derivative.

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

Theory:

Index Sequential File:

This is basically a mixture of sequential and indexed file organization 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 organization gives the index a tree structure. Obviously this type of file organization will require a direct access device, such as a hard disk.

Indexed sequential file organization is very useful where records are often retrieved randomly and are also processed in (sequential) key order. Banks may use this organization 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.

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

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.

PROGRAM:

```
#include <iostream>

#include <fstream>

#include <string>

using namespace std;

typedef struct seq_file
{
    int id;

    char name[20], desg[20];

    long int sal;
} record;

typedef struct ind_file
{
    int id;
```


NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
}index;

class file

{
    record data;

    index info;

public:

    void get_data()
    {
        cout<<"Enter id: ";

        cin>>data.id;

        cout<<"Enter name: ";

        cin>>data.name;

        cout<<"Enter designation: ";

        cin>>data.desg;

        cout<<"Enter salary: ";

        cin>>data.sal;

        info.id=data.id;
    }

    void add()
    {
        fstream out1;

        fstream out2;

        out1.open("pos.txt",ios::app);

        out2.open("rec.txt",ios::app);
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
    get_data();

    out2.write((char*)&data,sizeof(data));

    out1.write((char*)&info,sizeof(info));

    out1.close();

    out2.close();

}

void search_rec(int id)

{

    int pos=0,loc=-1;

    fstream out1;

    fstream out2;

    out1.open("pos.txt");

    out2.open("rec.txt");

    loc=sizeof(info)*pos;

    out2.seekg(loc,ios::beg);

    for(pos=0;out2.read((char*)&info,sizeof(info));pos++)

    {

        loc=sizeof(info)*pos;

        out2.seekg(loc,ios::beg);

        out2.read((char*)&info,sizeof(info));

        if(info.id==id)

        {

            break;

        }

    }

}
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
    }

    if(loc== -1)

    {

        cout<<"Record not found\n";

    }

    else

    {

        pos--;

        pos=sizeof(data)*pos;

        out1.seekg(pos,ios::beg);

        out1.read((char*)&data,sizeof(data));

        cout<<"Record found\n";

        cout<<data.id<<"\t"<<data.name<<"\t"<<data.desg<<"\t"<<data.sal<<endl;

    }

    out1.close();

    out2.close();

}

};

int main()

{

    char r;

    do

    {

        char op;
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
file f;

do
{
    int c;

    cout<<"\n=====Menu=====\\n";

    cout<<"1] Add record\\n2] Search record\\n3] Delete record\\n";

    cout<<"_____\\n";

    cout<<"Enter your choice: ";

    cin>>c;

    switch(c)
    {
        case 1: {
            f.add();
        }

        break;

        case 2: {
            int id;

            cout<<"Enter id to search: ";

            cin>>id;

            f.search_rec(id);
        }

        break;

        case 3: {
```

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062

```
        }

        break;

    case 4: {

        }

        break;

    default: cout<<"Error 404.....page not found\n";

    }

    cout<<"Do you wish to continue(y/n): ";

    cin>>op;

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

    cout<<"Test pass(y/n): ";

    cin>>r;

    }while(r=='n' || r=='N');

    cout<<"*****\n";

    cout<<"*   Thank You!   *\n";

    cout<<"*****\n";

    return 0;

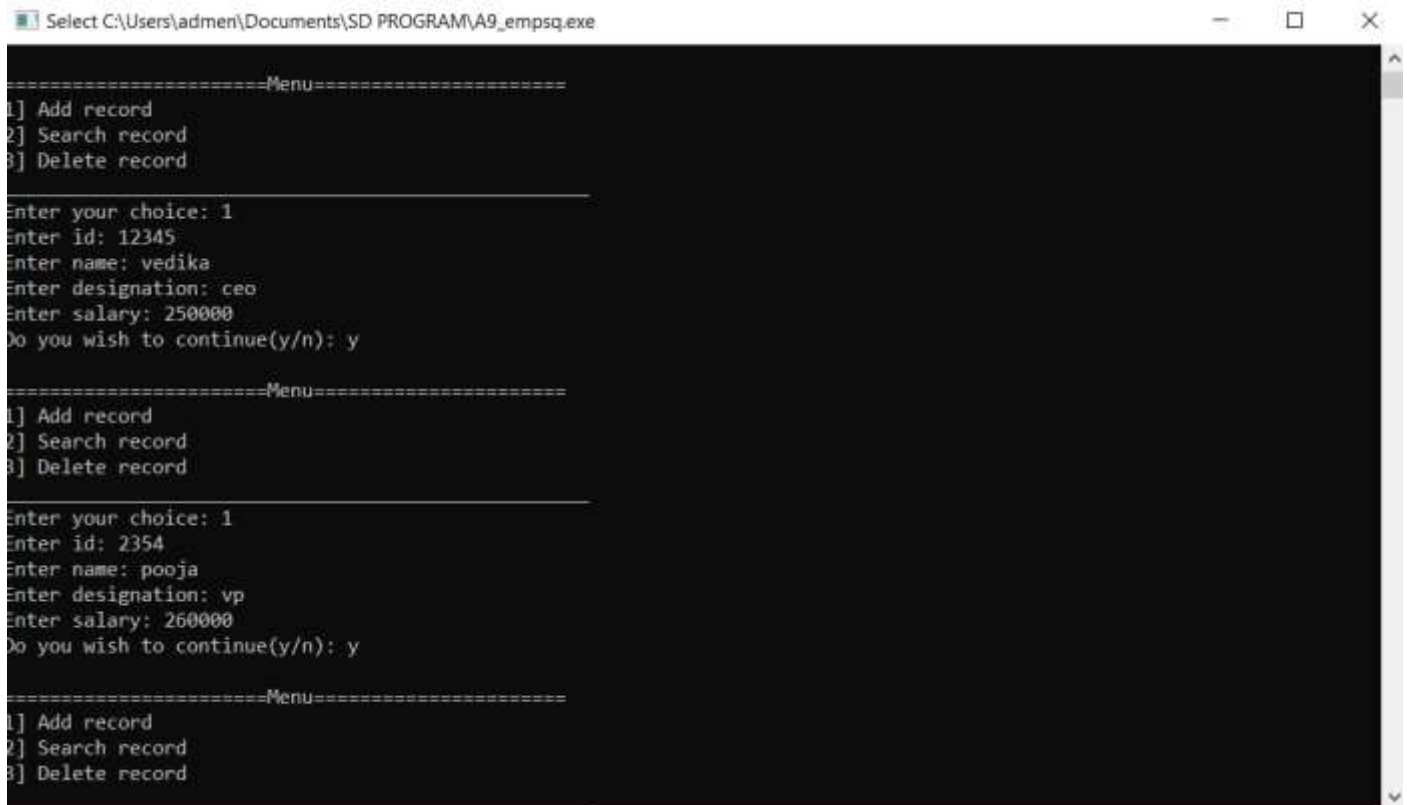
}
```

OUTPUT:

NAME: Ashish Khurkhuriya

Gr.No: u1610120

Roll. No: 223062



```
Select C:\Users\admin\Documents\SD PROGRAM\A9_empsq.exe

=====Menu=====
1] Add record
2] Search record
3] Delete record

Enter your choice: 1
Enter id: 12345
Enter name: vedika
Enter designation: ceo
Enter salary: 250000
Do you wish to continue(y/n): y

=====Menu=====
1] Add record
2] Search record
3] Delete record

Enter your choice: 1
Enter id: 2354
Enter name: pooja
Enter designation: vp
Enter salary: 260000
Do you wish to continue(y/n): y

=====Menu=====
1] Add record
2] Search record
3] Delete record
```

Conclusion:

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

NAME:Ashish Khurkhuriya

Gr.No: u1610120

Roll. No:223062