Assignment Number 1

Aim : To create ADT that implements the set concept.

1.Insert Element.
2.Remove Element.
3.Check Element.
4.Size of Set.
5.Intersection of Set.
6.Union.
7.Difference.
8.Subset.

Objectve : To implement set concepts.

Theory :

Union

Union of the sets A and B, denoted by A ∪ B, is the set of distinct element belongs to set A or set B, or both.

Intersection

The intersection of the sets A and B, denoted by A ∩ B, is the set of elements belongs to both A and B i.e. set of the common element in A and B.

Set Difference

Difference between sets is denoted by 'A – B', is the set containing elements of set A but not in B. i.e all elements of A except the element of B.

Algorithm :

Union*:*

1) Initialize union U as empty.
2) Copy all elements of first array to U.

3) Do following for every element x of second array:
…..a) If x is not present in first array, then copy x to U.
4) Return U.

Intersection:
1) Initialize intersection I as empty.
2) Do following for every element x of first array
…..a) If x is present in second array, then copy x to I.
4) Return I.

Program :

```
// Assignment 1_Set Operations using array

#include<iostream>
using namespace std;

class set
{
        int *p=NULL;
        int *q=NULL;
        int n;int m;
        public:
        void create();
        void insert();
        void remove();
        void contains();
        void size();
        void intersection();
        void uni();
        void diff();
        bool subset();
};
```

```
void set::create()
{
	int i;
	cout<<"How many elements do you want to enter in 1st
set?"<<endl;
	cin>>n;
	p=new int[n];
	cout<<"Enter elements"<<endl;
	for (i=0;i<n;i++)
	cin>>p[i];

	int j;
	cout<<"How many elements do you want to enter in 2nd
set?"<<endl;
	cin>>m;
	q=new int[m];
	cout<<"Enter elements"<<endl;
	for (j=0;j<m;j++)
	cin>>q[j];
	/*cout<<"entered elements"<<endl;
	for (j=0;j<m;j++)
	cout<<q[j]<<endl;*/
}

void set::insert()
{
	int s,i,j,ele;
	cout<<"_____
_____"<<endl;
	cout<<"In which set do you want to enter the elements"<<endl;
	cin>>s;
	if(s==1)
```

```
        {
                cout<<"Enter element"<<endl;
                cin>>ele;
                p[n]=ele;
                n=n+1;
                cout<<"The set is"<<endl;
                for (i=0;i<n;i++)
                cout<<p[i]<<endl;
        }
        else
        {
                cout<<"Enter element"<<endl;
                cin>>ele;
                q[m]=ele;
                m=m+1;
                cout<<"The set is"<<endl;
                for (j=0;j<m;j++)
                cout<<q[j]<<endl;
        }
}

void set::remove()
{
        cout<<"_____
_____"<<endl;
        int ele,i,j,flag=0,k,l;
        int pos;
        cout<<"Enter element to be removed"<<endl;
        cin>>ele;
        for (i=0;i<n;i++)
        {
        if(p[i]==ele)
        {
```

```
            //flag=1;
            pos=i;
            p[i] = 0;
            for (k=pos;k<n;k++)
            {
                  p[k] = p[k+1];
            }
            n=n-1;
            cout<<"Elemets of set after deletion are"<<endl;
            for (i=0;i<n;i++)
            cout<<p[i]<<endl;
      }
      }

      if(flag!=1)
      {
            for (j=0;j<m;j++)
            if(q[j]==ele)
            {
                  pos=j;
                  q[j] = 0;
                  for (l=pos;l<m;l++)
                  {
                        q[l] = q[l+1];

                  }
                  m=m-1;
                  cout<<"Elemets of set after deletion are"<<endl;
                  for (j=0;j<m;j++)
                  cout<<q[j]<<endl;
            }

      }
```

```
}

void set::contains()
{
	cout<<"_____
_____"<<endl;
	int i,j,ele,flag=0;
	cout<<"Which element is to be checked?"<<endl;
	cin>>ele;
	for (i=0;i<n;i++)
	if(p[i]==ele)
	{
		flag=1;
		cout<<"Element found in 1st set"<<endl;
	}
	for (j=0;j<m;j++)
	if(q[j]==ele)
	{
		flag=1;
		cout<<"Element found in second set"<<endl;
	}
	if(flag!=1)
	cout<<"Element not found in any set"<<endl;
}

void set::size()
{
	cout<<"_____
_____"<<endl;
	int s,i,j;
	cout<<"Size of which set is to be checked?"<<endl;
	cin>>s;
```

```
        if(s==1)
        {
                cout<<"Size of set is "<<n<<endl;
        }
        else
        {
                cout<<"Size of set is "<<m<<endl;
        }
}

void set::intersection()
{
        cout<<"_____
_____"<<endl;
        int i,j,flag=0,no;
        for (i=0;i<n;i++)
        for (j=0;j<m;j++)
        {
                if(p[i]==q[j])
                {
                        flag=1;
                        cout<<p[i]<<endl;
                }
        }

        if(flag!=1)
                {
                        cout<<"No common elements found !!"<<endl;
                }
}

void set::uni()
{
```

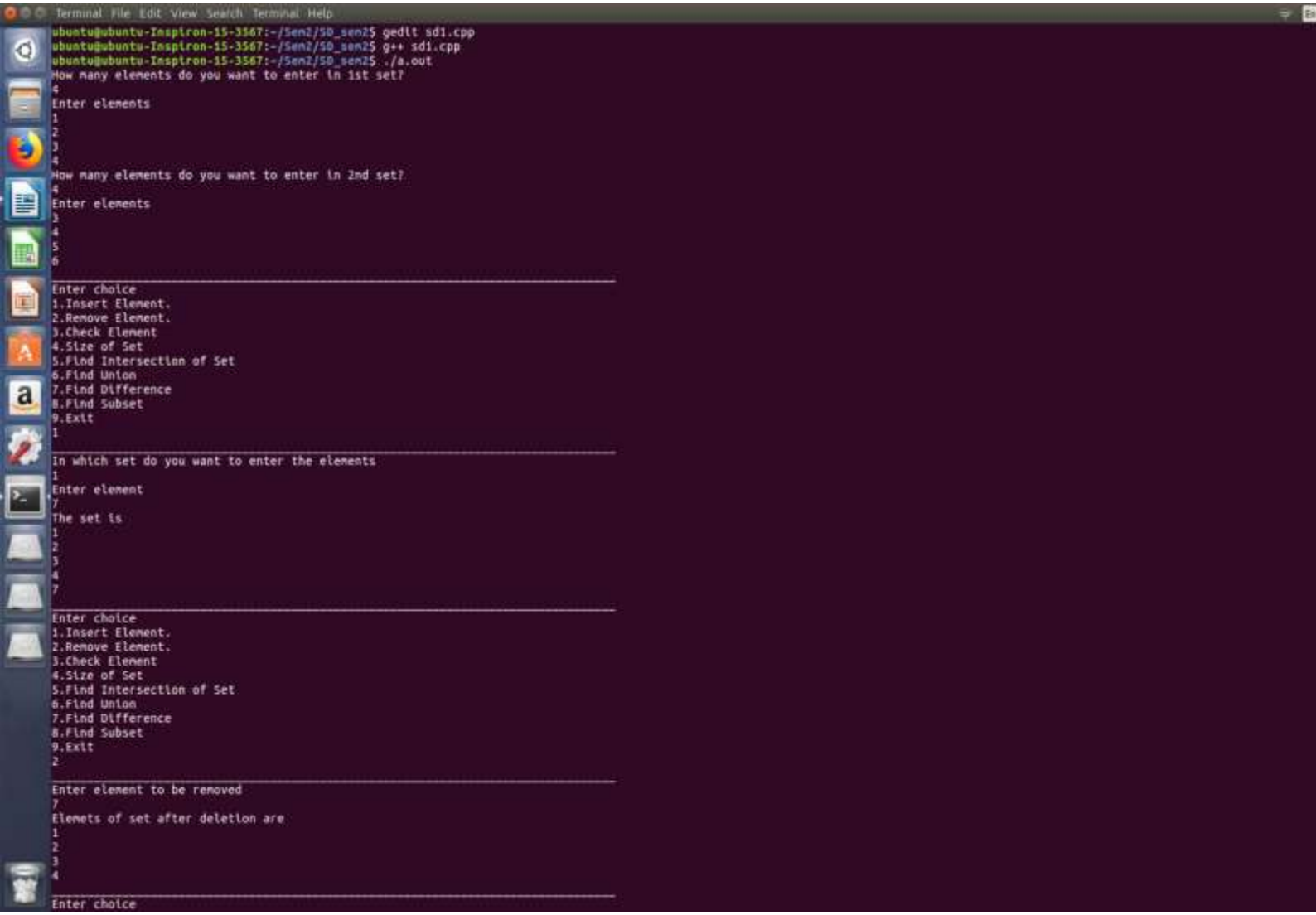
```
        cout<<"_____
_____"<<endl;
        int flag=0,k;
        int a[m+n];
        for(int i=0;i<n;i++)
        {
                a[i]=p[i];
        }
        k=n;
        for(int j=0;j<m;j++)
        {
                for(int i=0;i<n;i++)
                {
                        if(p[i]==q[j])
                        flag=1;
                }
                if(flag!=1)
                {
                        a[k++]=q[j];
                }
                else
                flag=0;
        }
        for(int i=0;i<k;i++)
        {
                cout<<a[i]<<endl;
        }

}

void set::diff()
{
```

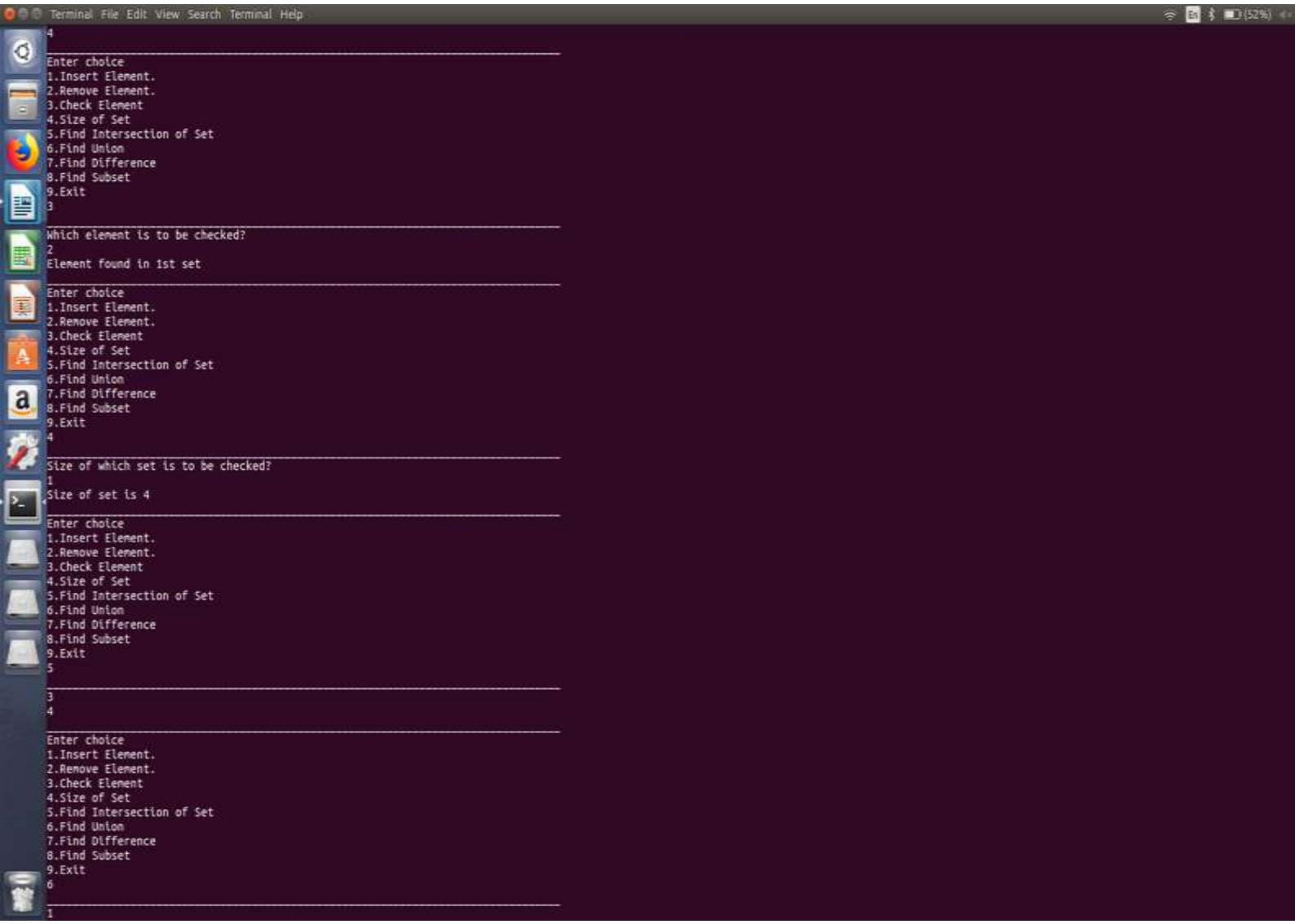
```
        cout<<"_____
_____"<<endl;
        int a[n];
        int k=0,flag=0;

        for(int i=0;i<n;i++)
        {
                for(int j=0;j<m;j++)
                {
                        if(p[i]==q[j])
                        flag=1;
                }
                if(flag!=1)
                {
                        a[k++]=p[i];
                }
                else
                flag=0;
        }

        for(int i=0;i<k;i++)
        {
                cout<<a[i]<<endl;
        }

}

bool set :: subset()
{

cout<<"_____
_____"<<endl;
   int i = 0;
```

```
   int j = 0;
   for (i = 0; i < m; i++)
   {
      for (j = 0; j < n; j++)
      {
         if(q[i] == p[j])
            break;
      }

      if (j == n)
      return 0;

   }

   return 1;
}

int main()
{

        set s;
        s.create();
        int ch;
        do
        {
        cout<<"_____
_____"<<endl;
        cout<<"Enter choice"<<endl;
        cout<<"1.Insert Element."<<endl<<"2.Remove
Element."<<endl<<"3.Check Element"<<endl<<"4.Size of
Set"<<endl<<"5.Find Intersection of Set"<<endl<<"6.Find
Union"<<endl<<"7.Find Difference"<<endl<<"8.Find
Subset"<<endl<<"9.Exit"<<endl;
```

```
        cin>>ch;
        switch(ch)
        {
                case 1:s.insert();
                break;
                case 2:s.remove();
                break;
                case 3:s.contains();
                break;
                case 4:s.size();
                break;
                case 5:s.intersection();
                break;
                case 6:s.uni();
                break;
                case 7:s.diff();
                break;
                case 8:
                        int x;
                        x=s.subset();
                        if(x)
                        cout<<"2nd set is a subset of 1st set"<<endl;
                        else
                        cout<<"2nd set is a not a subset of 1st set"<<endl;
                break;
                case 9:
                break;
        }
        }while(ch!=9);

        return 0;
}
```

Output :


Terminal File Edit View Search Terminal Help
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2/SD_sem2$ gedit sd1.cpp
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2/SD_sem2$ g++ sd1.cpp
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2/SD_sem2$ ./a.out
How many elements do you want to enter in 1st set?
4
Enter elements
1
2
3
4
How many elements do you want to enter in 2nd set?
4
Enter elements
3
4
5
6
Enter choice
1.Insert Element.
2.Remove Element.
3.Check Element
4.Size of Set
5.Find Intersection of Set
6.Find Union
7.Find Difference
8.Find Subset
9.Exit
1
In which set do you want to enter the elements
1
Enter element
7
The set is
1
2
3
4
7
Enter choice
1.Insert Element.
2.Remove Element.
3.Check Element
4.Size of Set
5.Find Intersection of Set
6.Find Union
7.Find Difference
8.Find Subset
9.Exit
2
Enter element to be removed
7
Elemets of set after deletion are
1
2
3
4
Enter choice



Conclusion : Thus we have implemented concepts of sets.

Assignment Number 2

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

Objectve : To create threaded binary tree and perform inorder traversal.

Theory :

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).

There are two types of threaded binary trees.
Single Threaded: Where a NULL right pointers is made to point to the in-order successor (if successor exists)
Double Threaded: Where both left and right NULL pointers are made to point to inorder predecessor and inorder successor respectively. The predecessor threads are useful for reverse inorder traversal and postorder traversal.
The threads are also useful for fast accessing ancestors of a node.

The idea of threaded binary trees is to make inorder traversal faster and do it without stack and without recursion.

Algorithm :

Let **tmp be the newly inserted node**. There can be three cases during insertion:
Case 1: Insertion in empty tree
Both left and right pointers of tmp will be set to NULL and new node becomes the root.

```
root = tmp;

tmp -> left = NULL;

tmp -> right = NULL;
```

Case 2: When new node inserted as the left child
After inserting the node at its proper place we have to make its left and right threads points to inorder predecessor and successor respectively. The node which was inorder successor. So the left and right threads of the new node will be-

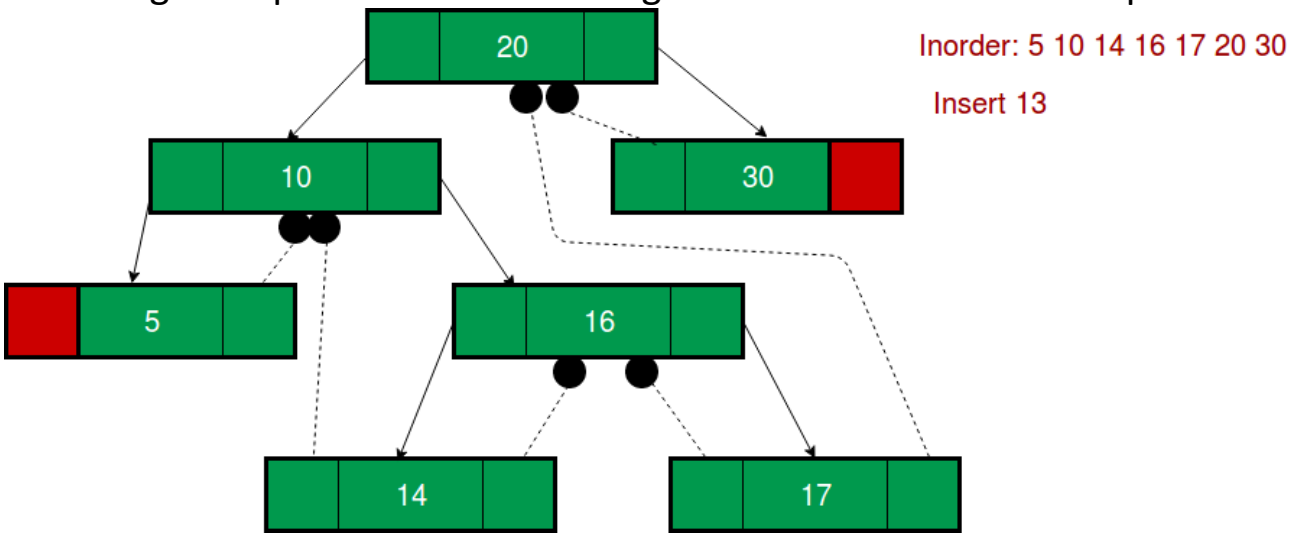
```
tmp -> left = par ->left;
```

```
tmp -> right = par;
```

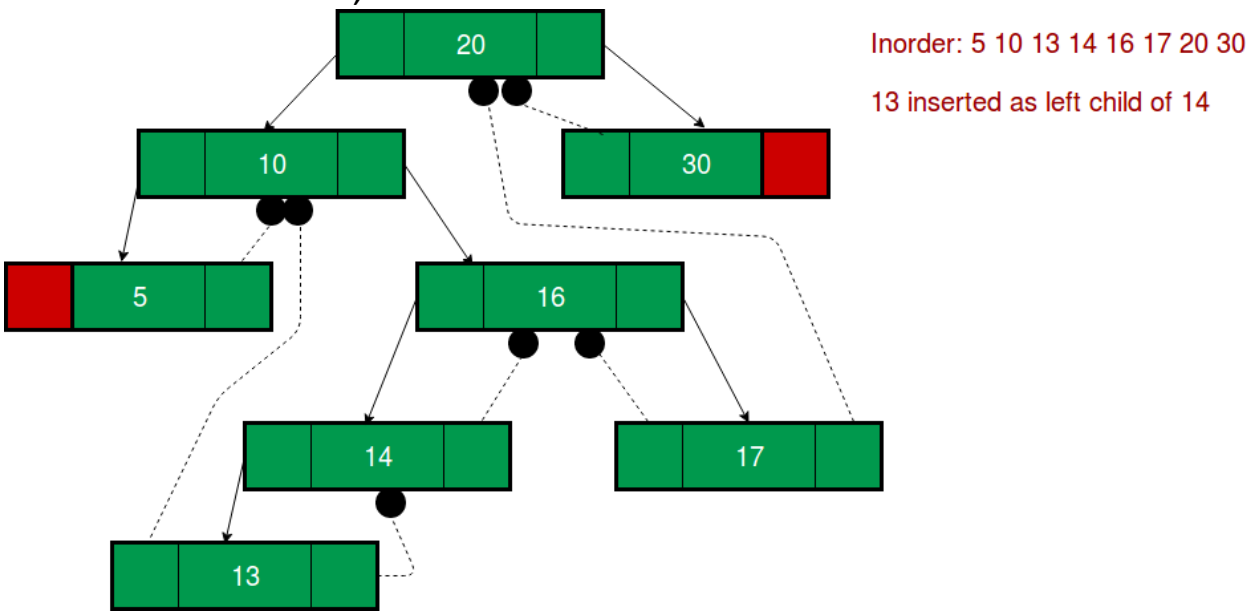
Before insertion, the left pointer of parent was a thread, but after insertion it will be a link pointing to the new node.

```
par ->lthread = false;

par -> left = temp;
```

Following example show a node being inserted as left child of its parent.



After insertion of 13,



Predecessor of 14 becomes the predecessor of 13, so left thread of 13 points to 10.

Successor of 13 is 14, so right thread of 13 points to left child which is 13. Left pointer of 14 is not a thread now, it points to left child which is 13.

Case 3: When new node is inserted as the right child

The parent of tmp is its inorder predecessor. The node which was inorder successor of the parent is now the inorder successor of this node tmp. So the left and right threads of the new node will be-

```
tmp -> left = par;
tmp -> right = par -> right;
```

Before insertion, the right pointer of parent was a thread, but after insertion it will be a link pointing to the new node.

```
par ->rthread = false;
par -> right = tmp;
```

Program :

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

class tbt_node
{
        int data;
        bool lth,rth;
        tbt_node *lptr;
        tbt_node *rptr;
        friend class tbt;
}*root,*dummy,a;

class tbt
{
        public:
```

```
        tbt()
        {
                a.lptr=a.rptr=NULL;
        }
        void create(int,int,tbt_node *);
        void insert(tbt_node*,tbt_node*);
        void inorder(tbt_node*);
};

void tbt::create(int j,int e,tbt_node *nn)
{

        for(j=0;j<e;j++)
        {
                nn=new tbt_node();
                nn->lth=nn->rth=1;
                cout<<"Enter data"<<endl;
                cin>>nn->data;

                if(root==NULL)
                {
                        root=nn;
                        dummy=new tbt_node();
                        dummy->data=999;
                        dummy->lptr=root;
                        dummy->rptr=dummy;
                        dummy->lth=dummy->rth=1;
                        root->lptr=root->rptr=dummy;
                }
                else
                {
                        insert(root,nn);
                }
```

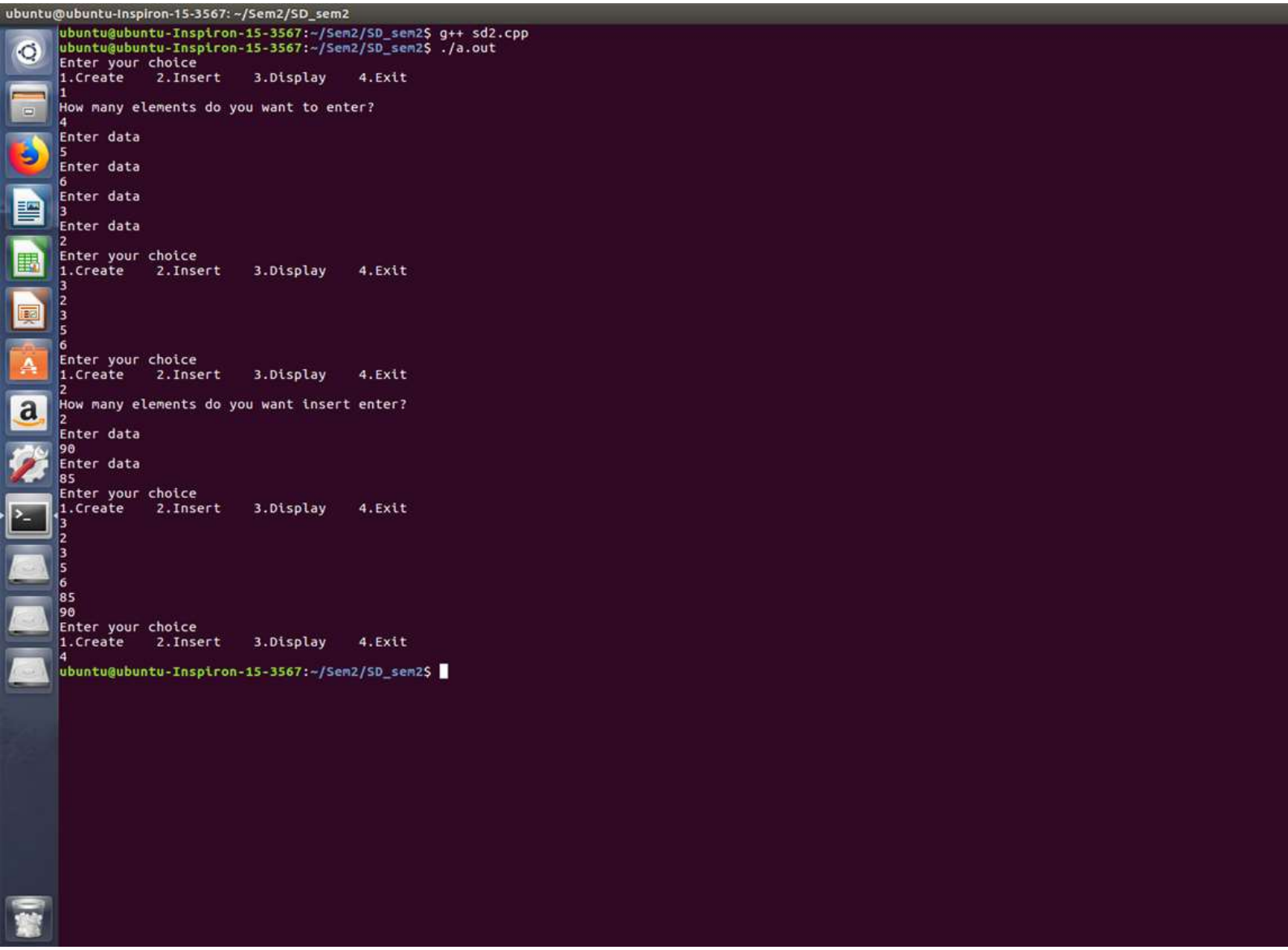
```
        }
}

void tbt::insert(tbt_node *temp,tbt_node *nn)
{
        if(temp->data>nn->data)
        {
                if(temp->lth==1)
                {
                        nn->lptr=temp->lptr;
                        temp->lptr=nn;
                        nn->rptr=temp;
                        temp->lth=0;
                }
                else
                {
                        insert(temp->lptr,nn);
                }
        }
        else if(temp->data<nn->data)
        {
                if(temp->rth==1)
                {
                        nn->rptr=temp->rptr;
                        temp->rptr=nn;
                        nn->lptr=temp;
                        temp->rth=0;
                }
                else
                {
                        insert(temp->rptr,nn);
                }
        }
```

```
        else
        {
                cout<<"Data already exists"<<endl;
                return;
        }
}

void tbt::inorder(tbt_node *temp)
{
        while(temp!=dummy)
        {
                while(temp->lth==0)
                {
                        temp=temp->lptr;
                }
                cout<<temp->data<<endl;
                while(temp->rth==1)
                {
                        temp=temp->rptr;
                        if(temp==dummy)
                        return;
                        cout<<temp->data<<endl;
                }
                temp=temp->rptr;
        }
}

int main()
{
        tbt t;
        tbt_node a;
        //t.create();
        //t.inorder(root);
```

```
        int ele;
        int i=0;
        tbt_node *nn,*temp;

        int ch;
        do
        {
                cout<<"Enter your choice"<<endl;
                cout<<"1.Create    2.Insert    3.Display    4.Exit"<<endl;
                cin>>ch;

                switch(ch)
                {
                        case 1: cout<<"How many elements do you want to
enter?"<<endl;
                                cin>>ele;
                                t.create(i,ele,nn);
                        break;

                        case 2: cout<<"How many elements do you want insert
enter?"<<endl;
                                cin>>ele;
                                t.create(i,ele,nn);
                        break;

                        case 3: t.inorder(root);
                        break;

                        case 4:
                        break;
                }
        }while(ch!=4);
        return 0;
```

}

Output :



Conclusion : Thus we have constructed a threaded binary tree.

Assignment Number 3

Aim : Represent a Graph using adjacency matrix.

Objectve : To learn adjacency list representation of graph.

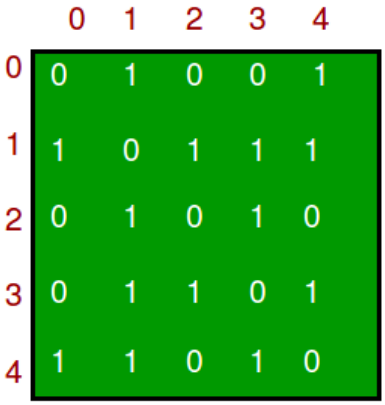
Theory : Graph is a data structure that consists of following two components:
1. A finite set of vertices also called as nodes.
2. A finite set of ordered pair of the form (u, v) called as edge.
Graphs are used to represent many real-life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network.

Following two are the most commonly used representations of a graph.
1. Adjacency Matrix
2. Adjacency List

There are other representations also like, Incidence Matrix and Incidence List. The choice of the graph representation is situation specific. It totally depends on the type of operations to be performed and ease of use.

Adjacency Matrix:

Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j. Adjacency matrix for undirected graph is always symmetric. Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.
The adjacency matrix for the above example graph is:

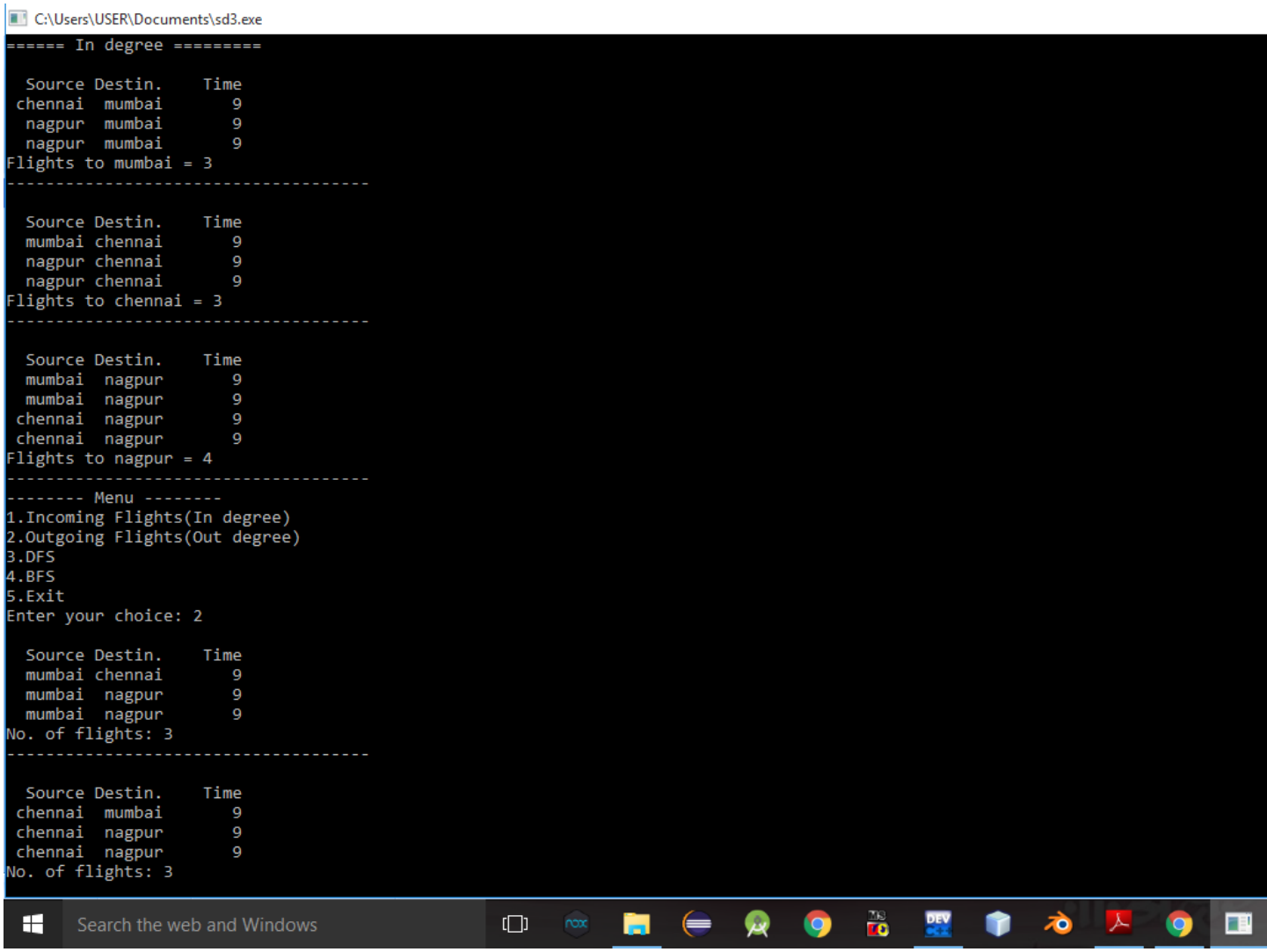


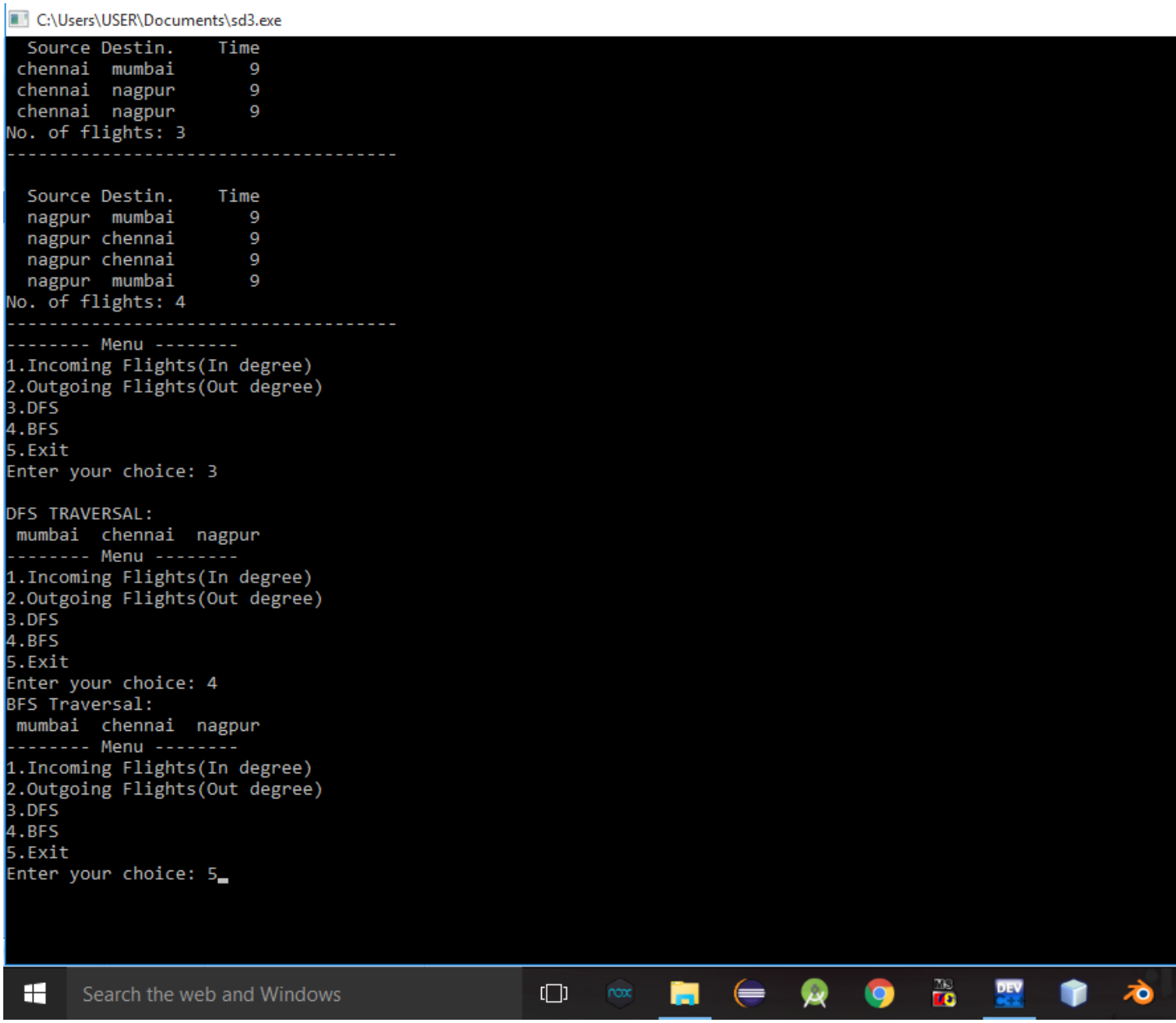
Program :

```
#include<iostream>
#define MAX 10
using namespace std;
class airport
{
        string city[MAX];
        int distance[10][10];
public :
int n;
airport();
voidread_city();
voidshow_graph();
};
airport::airport()
{
        n=0;
        for(inti=0;i<MAX;i++)
        {
                for(int j=0;j<MAX;j++)
                        distance[i][j]=0;
        }
}
void airport::read_city()
{
        int k;
        cout<<"\nEnter the no. of cities: " ;
        cin>>n;
        cout<<"Enter city name:\n";
        for(int k=0;k<n;k++)
```

```
        {
        cout<<k+1<<"] ";
                cin>>city[k];
        }
        for(inti=0;i<n;i++)
        {
                for(int j=i+1 ; j<n ; j++)
                {
                        cout<<"\nEnter Distance between "<<city[i]<<" to "<<city[j]<<": ";
                        cin>>distance[i][j];
                        distance[j][i]=distance[i][j];
                }
        }
}
void airport::show_graph()
{
cout<<"\t";
for(int k=0;k<n;k++)
   {
cout<<city[k]<<"\t";
   }
cout<<endl;
for(inti=0;i<n;i++)
   {
cout<<city[i]<<"\t";
for(int j=0;j<n;j++)
      {
cout<<distance[i][j]<<"\t";
      }
cout<<endl;
   }
}
int main()
{
        airportobj;
        obj.read_city();
        obj.show_graph();
}
```

OUTPUT:-


C:\Users\USER\Documents\sd3.exe
====== In degree =========
Source Destin. Time
chennai mumbai 9
nagpur mumbai 9
nagpur mumbai 9
Flights to mumbai = 3
Source Destin. Time
mumbai chennai 9
nagpur chennai 9
nagpur chennai 9
Flights to chennai = 3
Source Destin. Time
mumbai nagpur 9
mumbai nagpur 9
chennai nagpur 9
chennai nagpur 9
Flights to nagpur = 4
-------- Menu --------
1.Incoming Flights(In degree)
2.Outgoing Flights(Out degree)
3.DFS
4.BFS
5.Exit
Enter your choice: 2
Source Destin. Time
mumbai chennai 9
mumbai nagpur 9
mumbai nagpur 9
No. of flights: 3
Source Destin. Time
chennai mumbai 9
chennai nagpur 9
chennai nagpur 9
No. of flights: 3
Search the web and Windows



Conclusion : We saw all the algorithms the STL offers to operate on sets, that are collections of sorted elements, in the general sense.

Assignment Number 4

Aim :

For a weighted graph G, find the minimum spanning tree using Prim's Algorithm.

Objectve : To implement Prim's Algorithm.

Theory :

A group of edges that connects two set of vertices in a graph is called cut in graph theory. *So, at every step of Prim's algorithm, we find a cut (of two sets, one contains the vertices already included in MST and other contains rest of the verices), pick the minimum weight edge from the cut and include this vertex to MST Set (the set that contains already included vertices).*

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*

Program :

```
//Assignment 4
//Prims Algorithm
```

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

int INT_MAX=999;
class prims
{
	public:

	prims()
	{}
	int** build(int);
	void print(int[],int,int**,int);
	int minKey(int[],bool[], int);
	void Prims_Algo(int**,int);
};

int** prims::build(int V)
{
	int i,j,cost,e;
	int** graph=new int*[V];


	for(int i=0;i<V;i++)
	{
		graph[i]=new int [V];
		for(int j=0;j<V;j++)
		{
		graph[i][j]=0;
		}

	}

	cout<<"Enter number of edges"<<endl;
```

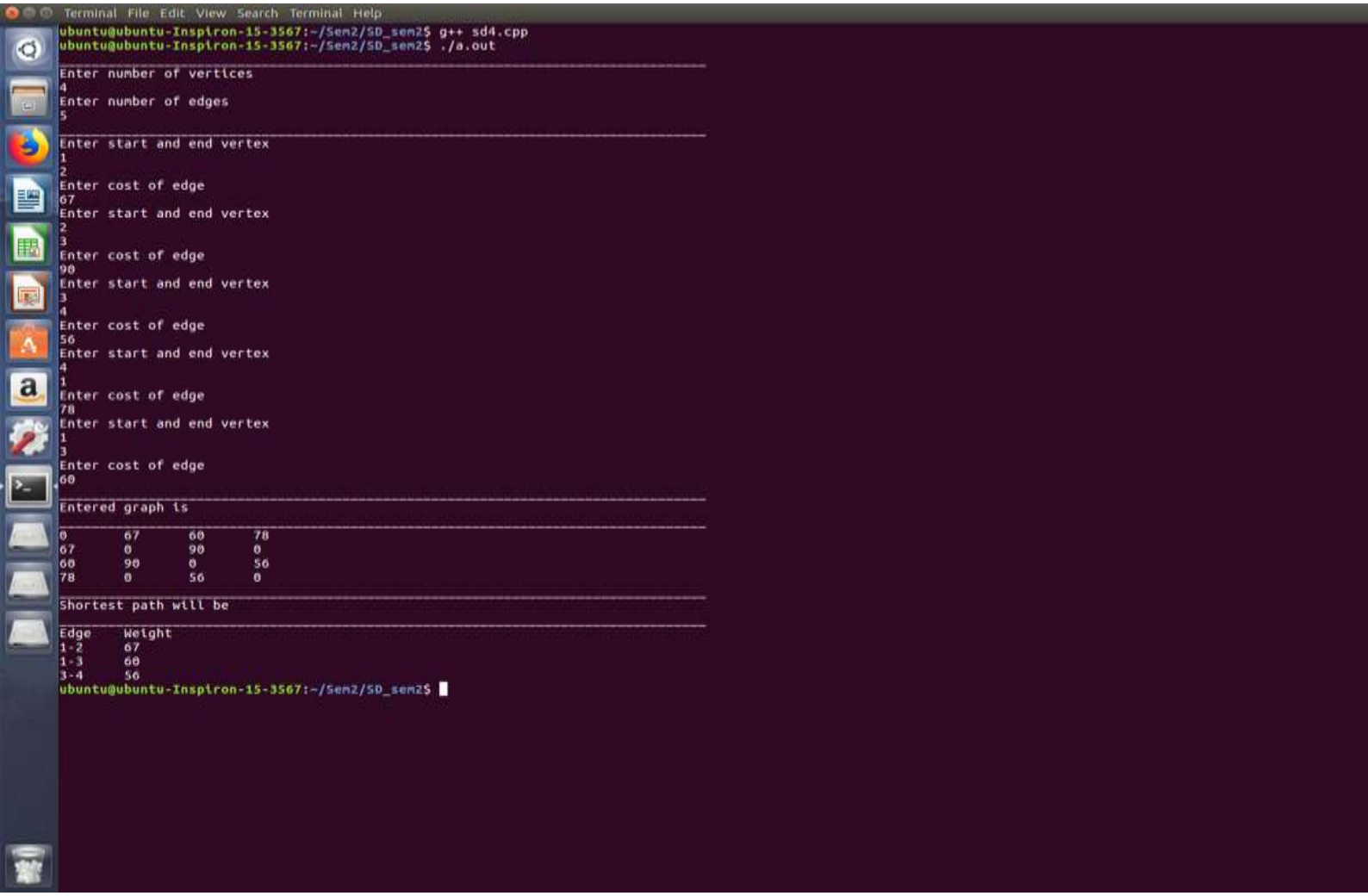
```
        cin>>e;
        cout<<"_____
_____"<<endl;

        for(int k=1;k<=e;k++)
        {
                cout<<"Enter start and end vertex"<<endl;
                cin>>i>>j;


                cout<<"Enter cost of edge"<<endl;
                cin>>cost;
                graph[i-1][j-1]=cost;
                graph[j-1][i-1]=cost;
        }

        cout<<"_____
_____"<<endl;
        cout<<"Entered graph is "<<endl;
        cout<<"_____
_____"<<endl;
        for(int k=0;k<V;k++)
        {
                for(int l=0;l<V;l++)
                {
                cout<<graph[k][l]<<"\t";
                }
                cout<<"\n";
        }

        return graph;
}
```

```
void prims::print(int parent[], int n,int** g,int V)
{
	cout<<"_____
_____"<<endl;
	cout<<"Shortest path will be"<<endl;
	cout<<"_____
_____"<<endl;
	cout<<"Edge \tWeight\n";
	for (int i = 1; i < V; i++)
	cout<< parent[i]+1<<"-"<< i+1<<"\t"<< g[i][parent[i]]<<endl;

}

int prims::minKey(int key[], bool mstSet[], int V)
{
	// Initialize min value

	int min = INT_MAX;
	int min_index;

	for (int v = 0; v < V; v++)
	if (mstSet[v] == false && key[v] < min)
		min = key[v], min_index = v;

	return min_index;

}

void prims::Prims_Algo(int** g,int V)
{
	// Array to store constructed MST
	int parent[V];
	// Key values used to pick minimum weight
```

```
int key[V];
// To represent set of vertices not yet included in MST
bool mstSet[V];


for (int i = 0; i < V; i++)
{
        key[i] = INT_MAX;
         mstSet[i] = false;
}

// Always include first 1st vertex in MST.
// Make key 0 so that this vertex is picked as first vertex.
key[0] = 0;
parent[0] = -1; // First node is always root of MST

// The MST will have V vertices
for (int count = 0; count < V-1; count++)
{
        // Pick the minimum key vertex from the
        // set of vertices not yet included in MST
        int u = minKey(key, mstSet,V);

        // Add the picked vertex to the MST Set
        mstSet[u] = true;

        // Update key value and parent index of
        // the adjacent vertices of the picked vertex.
        // Consider only those vertices which are not
        // yet included in MST
        for (int z = 0; z < V; z++)

        // graph[u][v] is non zero only for adjacent vertices of m
```

```
            // mstSet[v] is false for vertices not yet included in MST
            // Update the key only if graph[u][v] is smaller than key[v]
            if (g[u][z] && mstSet[z] == false && g[u][z] < key[z])
            {
                parent[z] = u;
                 key[z] = g[u][z];
            }
        }

        // print the constructed MST
        print(parent, V, g,V);

}

int main()
{
        prims p;
        int V;
        cout<<"_____
_____"<<endl;
        cout<<"Enter number of vertices"<<endl;
        cin>>V;
        //p.build(V);
        int **g=p.build(V);
        p.Prims_Algo(g,V);
        return 0;
}
```

Output :



Conclusion : Thus we implemented Prim's Algorithm.

Assignment Number 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.

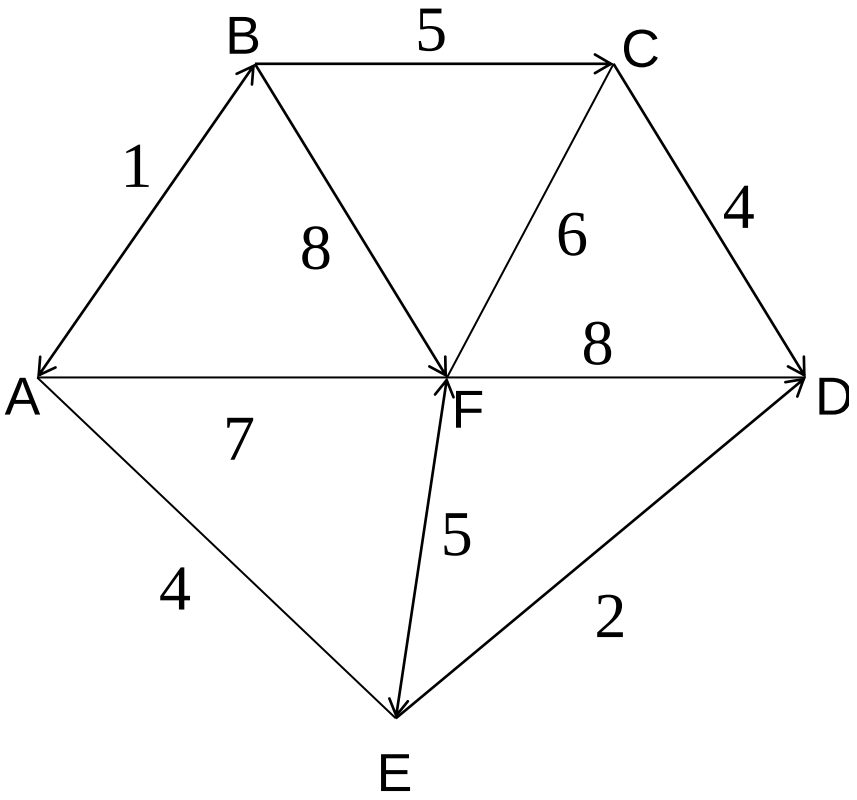
Objective:

To understand the concept of minimum spanning tree and finding the minimum cost of tree using Kruskals algorithm.

Theory:

A spanning tree of the graph is a connected (if there is at least one path between every pair of vertices in a graph) subgraph in which there are no cycle. Suppose you have a connected undirected graph with a weight (or cost) associated with each edge. The cost of a spanning tree would be the sum of the costs of its edges. A minimum-cost spanning tree is a spanning tree that has the lowest cost. There are two basic algorithms for finding minimum-cost spanning trees: 1. Prim's Algorithm 2. Kruskal's Algorithm . Kruskals's algorithm: It tarts with no nodes or edges in the spanning tree, and repeatedly add the cheapest edge that does not create a cycle.
Steps of Kruskal's Algorithm to find minimum spanning tree:

1. Select the shortest edge in a network
2. Select the next shortest edge which does not create a cycle
3. Repeat step 2 untill spanning tree has n-1 edges.



Example:

The solution is

AB 1

ED 2

CD 4

AE 4

EF 5

Total weight of tree: 16

Algorithm:

- Algorithm kruskal(G,V,E,T)

{

1.Sort E in increasing order of weight

2.let G=(V,E) and T=(A,B),A=V ,B is null set and let n =count(V)

3.Initialize n set ,each containing a different element of v.

4.while(|B|<n-1) do

begin

```
     e=<u,v>the shortest edge not yet considered
     U=Member(u)
      V=Member(v)
    if( Union(U,V))
        update in B and add the cost
     } }
  end
5.T is the minimum spanning tree
}
```

Program code:-

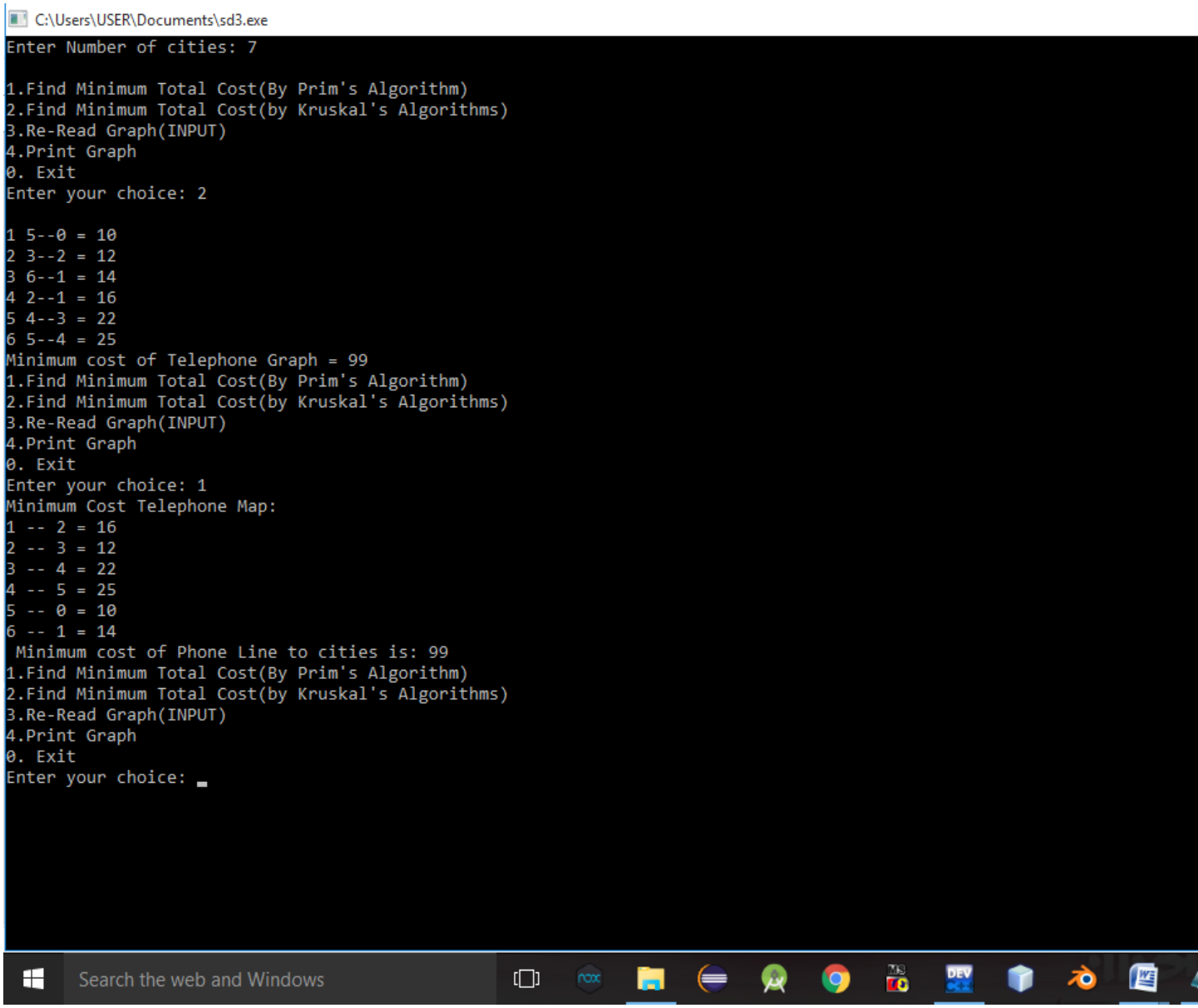
```
#include<iostream>
#define MAX 999;
using namespace std;

class kruskal
{
private:
   struct node
   {
      int v1,v2,cost;
   }G[20];
public:
   int edges,vertices;
   void create();
   void mincost();
   void input();
   int minimum(int);
};
int find (int v2,int parent[])
{
   while(parent[v2]!=v2)
   {
```

```
      v2=parent[v2];
   }
}
void uni(int i,int j,int parent[])
{
   if(i<j)
      parent[j]=i;
   else
      parent[i]=j;
}
void kruskal::input()
{
   cout<<"enter number of companies"<<endl;
   cin>>vertices;
   cout<<"enter number of connection"<<endl;
   cin>>edges;
}
void kruskal::create()
{
   cout<<"\n enter edges in v1-v2 form and corresponding cost"<<endl;
   for(int k=0;k<edges;k++)
   {
     cin>>G[k].v1>>G[k].v2>>G[k].cost;
   }
}
int kruskal::minimum(int n)
{
   int i,small,pos;
   small=MAX;
   pos=-1;
   for(i=0;i<n;i++)
   {
      if(G[i].cost<small)
```

```
      {
         small=G[i].cost;
         pos=i;
      }
   }
   return pos;
}
void kruskal::mincost()
{
   int count,k,v1,v2,i,j,tree[10][10],pos,parent[10];
   int sum=0;
   count=0;
   k=0;
   for(i=0;i<vertices;i++)
      parent[i]=i;
   while(count!=vertices-1)
   {
   pos=minimum(edges);
   if(pos==-1)
      break;
   v1=G[pos].v1;
   v2=G[pos].v2;
   i=find(v1,parent);
   j=find(v2,parent);
   if(i!=j)
      {
      tree[k][0]=v1;
      tree[k][1]=v2;
      k++;
      count++;
      sum=sum+G[pos].cost;
      uni(i,j,parent);
      }
```

```
   G[pos].cost=MAX;
   }
   if(count==vertices-1)
   {
      cout<<"spanning tree is"<<endl;
      for(i=0;i<vertices-1;i++)
      {
         cout<<tree[i][0]<<"-"<<tree[i][1]<<endl;
      }
      cout<<"cost required to set cables"<<sum<<endl;
   }
   else
   {
      cout<<"connection can't be set up"<<endl;
   }
}
int main()
{
   kruskal k;
   k.input();
   k.create();
   k.mincost();
}
```

Output:-



Conclusion: Kruskal's algorithm can be shown to run in O(**E log E**) time, where E is the number of edges in the graph. Thus we have connected all the offices with a total minimum cost using kruskal's algorithm.

Assignment Number 6

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

Objectve : To use heap data structure.

Theory :

Heap Property
If A is a parent node of B, then the key (the value) of node A is ordered with respect to the key of node B with the same ordering applying across the heap.

MAX HEAP definition:

Complete (Binary) tree with the property that the **value of each node** is at least as large as the value of its children (i.e. >= value of its children)

MIN HEAP definition:

Complete (Binary) tree with the property that the **value of each node** is at most as large as the value of its children (i.e. <= value of its children)

Algorithm :

MAX-HEAPIFY(A, i, n)

1. l ← LEFT(i)
2. r ← RIGHT(i)
3. if l ≤ n and A[l] > A[i]
4. then largest ←l

5. else largest ←i
6. if r ≤ n and A[r] > A[largest]
7. then largest ←r
8. if largest ≠ i
9. then exchange A[i] ↔ A[largest]
10. MAX-HEAPIFY(A, largest, n)

HEAPSORT*(A)*

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

Program :

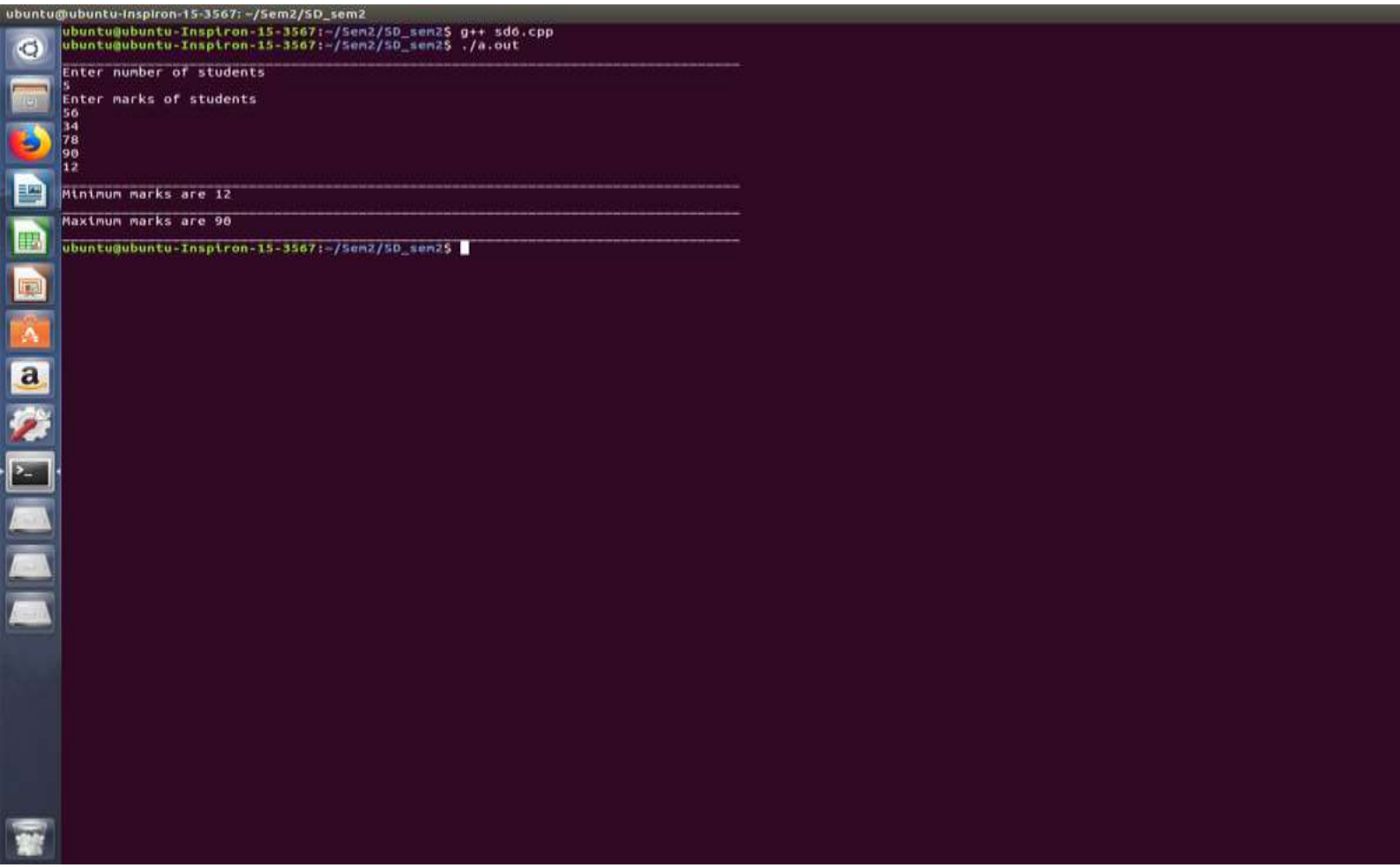
```
#include <iostream>
using namespace std;

class student
{

        public:
        student()
        {

        }
        void build(int[],int);
        void heapify(int [],int ,int);
        void sort(int[],int);
};

void student::build(int a[],int n)
{
```

```
        // Build heap (rearrange array)
        for (int i = n / 2 - 1; i >= 0; i--)
      heapify(a, i, n);

}

void student::heapify(int a[],int i,int n)
{
        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 && a[l] > a[largest])
      largest = l;

        // If right child is larger than largest so far
        if (r < n && a[r] > a[largest])
      largest = r;

        // If largest is not root
        if (largest != i)
        {
        swap(a[i], a[largest]);

        // Recursively heapify the affected sub-tree
        heapify(a, largest, n);
        }
}

void student::sort(int a[],int n)
{
```

```
        build(a,n);
        // One by one extract an element from heap
        for (int i=n-1; i>=0; i--)
        {
        // Move current root to end
        swap(a[0], a[i]);

        // call max heapify on the reduced heap
        heapify(a, 0, i);
        }
        /*cout << "Sorted array is"<<endl;
        for(int i=0;i<n;i++)
        cout<<a[i]<<endl;*/


        cout<<"_____
_____"<<endl;
        cout<<"Minimum marks are "<<a[0]<<endl;
        cout<<"_____
_____"<<endl;
        cout<<"Maximum marks are "<<a[n-1]<<endl;
        cout<<"_____
_____"<<endl;
}

int main()
{
        student s;
        int n;
        int *a=NULL;
```

```
        cout<<"_____
_____"<<endl;
        cout<<"Enter number of students"<<endl;
        cin>>n;
        a=new int[n];
        cout<<"Enter marks of students"<<endl;
        for(int i=0;i<n;i++)
        cin>>a[i];
        s.build(a,n);

        s.sort(a,n);

return 0;
}
```

Output :



Conclusion :

Thus we have stored information using heap data structure.

Assignment Number 7

Aim :

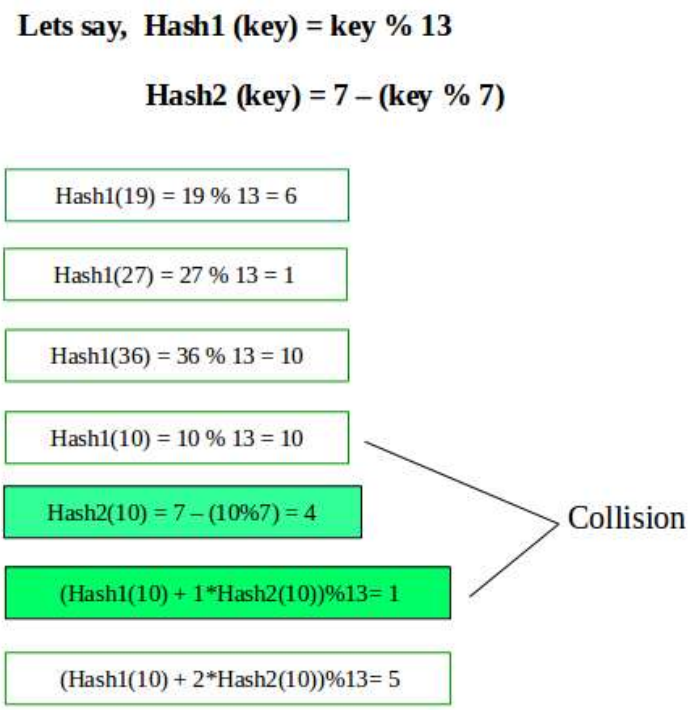
Insert the keys into a hash table of length m using open addressing using double hashing using h(k)=1+(k%(m-1)).

Objectve : To use hash tables to store data.

Theory :

A map is a relation between two sets.
We can define Map M as a set of pairs, where each pair is of the form (key, value), where for given a key, we can find a value using some kind of a “function” that maps keys to values.
The key for a given object can be calculated using a function called a hash function.
For example, an array is a Map where key is the index and value is the value at that index.
Hash table is a generalized idea of an array where key does not have to be an integer. We can have a name as a key, or for that matter any object as the key.
The trick is to find a hash function to compute an index so that an object can be stored at a specific location in a table such that it can easily be found.

Algorithm :



Program :

```
//Assignment 8
//Hashing

#include<iostream>
using namespace std;

class hash
{
        public:
        hash(){}
        bool isFull(int [],int);
        void calc_hash(int, int [], int);
        void store(int ,int [],int ,int );
        void display(int [],int);
};
```

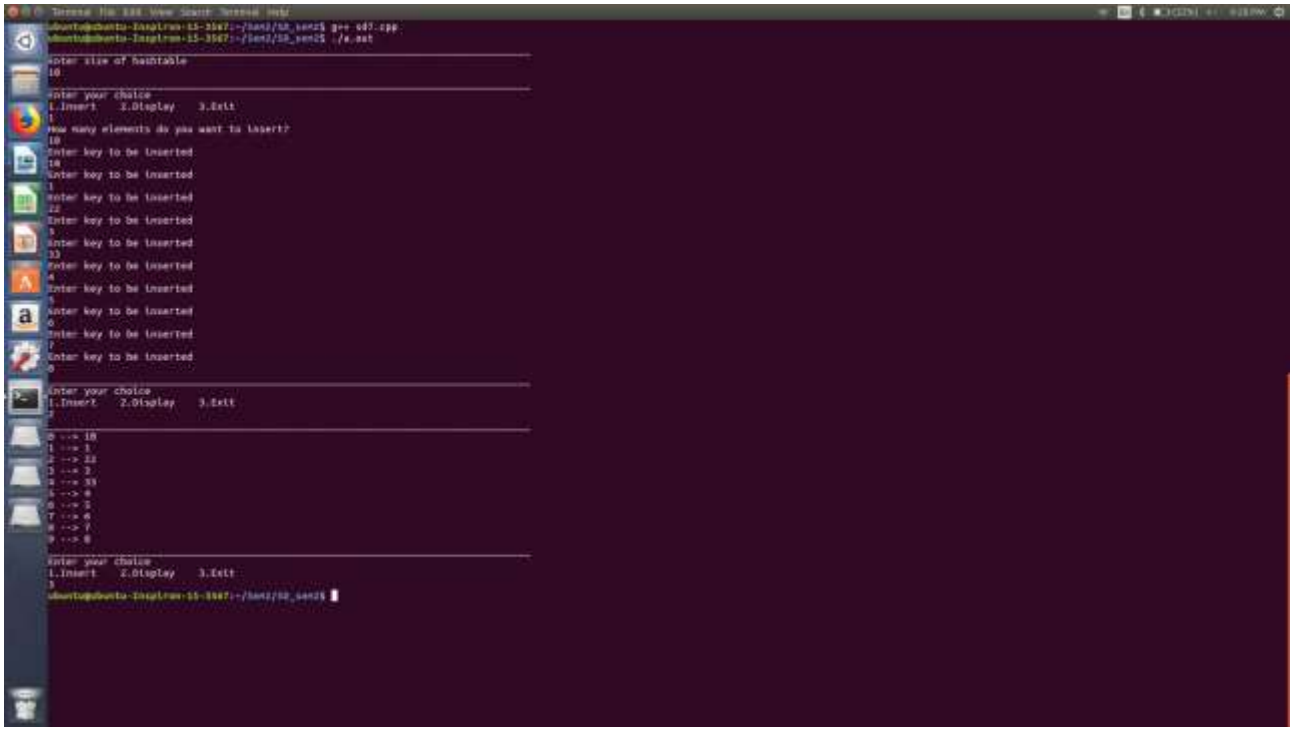
```
bool hash :: isFull(int hash_table[],int size)
{
	int i;
	for( i=0;i<size;i++)
	{
		if(hash_table[i]==-1)
		break;
	}


	if(i==size)
	return 1;
	else
	return 0;
}

void hash :: calc_hash(int key,int hash_table[],int size)
{
	int H;

		H=key%(size);
		if(hash_table[H]==-1)
		{
		store(key,hash_table,size,H);
		}
		else
		{
			H=1+(key%(size));
			store(key,hash_table,size,H);
		}


}
```

```
void hash :: store(int key,int hash_table[],int size,int index)
{
        hash_table[index]=key;
}

void hash :: display(int hash_table[],int size)
{
        cout<<"_____
_____"<<endl;
        for (int i = 0; i < size; i++)
       {
          if (hash_table[i] != -1)
             cout << i << " --> "
                 << hash_table[i] << endl;
          else
             cout << i << endl;
       }
}

int main()
{
        hash s;
        int size;
        cout<<"_____
_____"<<endl;
        cout<<"Enter size of hashtable"<<endl;
        cin>>size;
        int hash_table[size];
        for(int i=0;i<size;i++)
        {

                hash_table[i]=-1;
```

```
	}

	int ch,ele;
	int key,index;
	do
	{
	cout<<"_____
_____"<<endl;
	cout<<"Enter your choice"<<endl;
	cout<<"1.Insert    2.Display    3.Exit"<<endl;
	cin>>ch;
	switch(ch)
	{
	case 1:{
		cout<<"How many elements do you want to insert?"<<endl;
		cin>>ele;
		if(ele>size)
		{
			cout<<"Number of elements exceeding size"<<endl;
		}
		else
		{
		for(int i=0;i<ele;i++)
		{
		int x=s.isFull(hash_table,size);

		if(x==1)
		{
			cout<<"Hash table is full !"<<endl;
		}
		else
		{
```

```
                cout<<"Enter key to be inserted"<<endl;
                cin>>key;
                s.calc_hash(key,hash_table,size);
                }
                }
                }
                }
        break;
        case 2:
                s.display(hash_table,size);
        break;
        case 3:
        break;
        }
        }while(ch!=3);
        return 0;
}
```

Output :



Conclusion : Hence hashing is implemented.

Assignment Number 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

Objective:
To study different data structure concepts to implement this program.

Theory:

Input/output formatting

Writing to or reading from a file is similar to writing onto a terminal screen or reading from a keyboard. Differences are:

- File must be opened with an OPEN statement, in which the unit number and (optionally) the filename are given
- Subsequent writes (or reads) must refer to a known unit number (used for open)
- File should be closed at the end

File opening and closing

The syntax is:

```
OPEN([unit=]lunit,file='name' [,options])
```

```
CLOSE([unit=]lunit [,options])
```

For example:

```
OPEN(10, file='output.dat', status='new')

CLOSE(unit=10)
```

- The first parameter is the unit number and the keyword unit= can be omitted.
- The unit numbers 0,5 and 6 are predefined.
  - 0 is output for standard (system) error messages
  - 5 is for standard (user) input
  - 6 is for standard (user) output
  - These units are opened by default and should not be re-opened nor closed by users

Some options for opening a file:

- status: existence of the file ('old', 'new', 'replace', 'scratch', 'unknown')
- position: offset, where to start writing ('append')
- action: file operation mode ('write','read','readwrite')
- form: text or binary file ('formatted', 'unformatted')
- access: direct or sequential file access ('direct','sequential','stream')
- iostat: error indicator, (output) integer (non zero only upon an error)
- err: the label number to jump upon error
- recl: record length, (input) integer for direct access files only. Be careful, it can be in bytes or words...
-

Algorithm:
Program Code:

```
#include <iostream>
#include<fstream>
#include<cstring>
#include<iomanip>
using namespace std;

const int MAX=20;

class Student
{
 int rollno;
 char name[20],city[20];
 char div;
 int year;
        public:
         Student()
        {
          strcpy(name,"");
          strcpy(city,"");
          rollno=year=div=0;
        }
         Student(int rollno,char name[MAX],int year,char div,char city[MAX])
         {
          strcpy(this->name,name);
          strcpy(this->city,city);
          this->rollno=rollno;
          this->year=year;
          this->div=div;
         }
         int getRollNo()
         {
          return rollno;
```

```
        }
        void displayRecord()
        {


cout<<endl<<setw(5)<<rollno<<setw(20)<<name<<setw(5)<<year<<setw(
5)<<div<<setw(10)<<city;
        }
};

//==========File Operations ==========
class FileOperations
{
 fstream file;
        public:
         FileOperations(char* filename)
        {
        file.open(filename,ios::in|ios::out|ios::ate|ios::binary);
        }
         void insertRecord(int rollno, char name[MAX],int year, char div,char
city[MAX])
         {
          Student s1(rollno,name,year,div,city);
          file.seekp(0,ios::end);
          file.write((char *)&s1,sizeof(Student));
          file.clear();
         }
         void displayAll()
         {
          Student s1;
          file.seekg(0,ios::beg);
          while(file.read((char *)&s1, sizeof(Student)))
          {
```

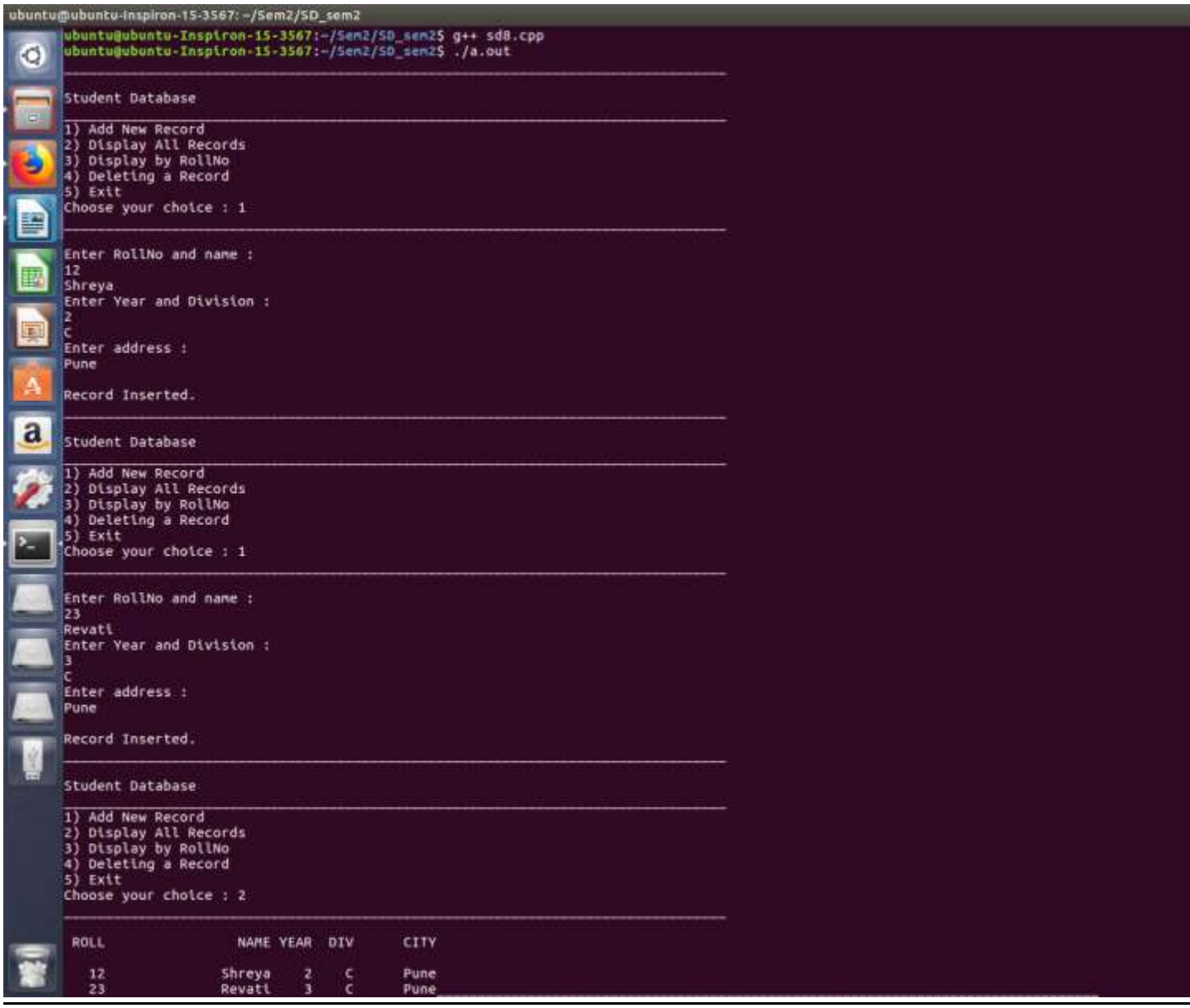
```
        s1.displayRecord();
       }
       file.clear();
      }
      void displayRecord(int rollNo)
      {
       Student s1;
       file.seekg(0,ios::beg);
       bool flag=false;
       while(file.read((char*)&s1,sizeof(Student)))
       {
        if(s1.getRollNo()==rollNo)
        {
         s1.displayRecord();
         flag=true;
         break;
        }
       }
       if(flag==false)
       {

cout<<"_____
_____"<<endl;
        cout<<"\nRecord of "<<rollNo<<"is not present."<<endl;
       }
       file.clear();
      }
      void deleteRecord(int rollno)
      {
       ofstream outFile("new.dat",ios::binary);
       file.seekg(0,ios::beg);
       bool flag=false;
       Student s1;
```

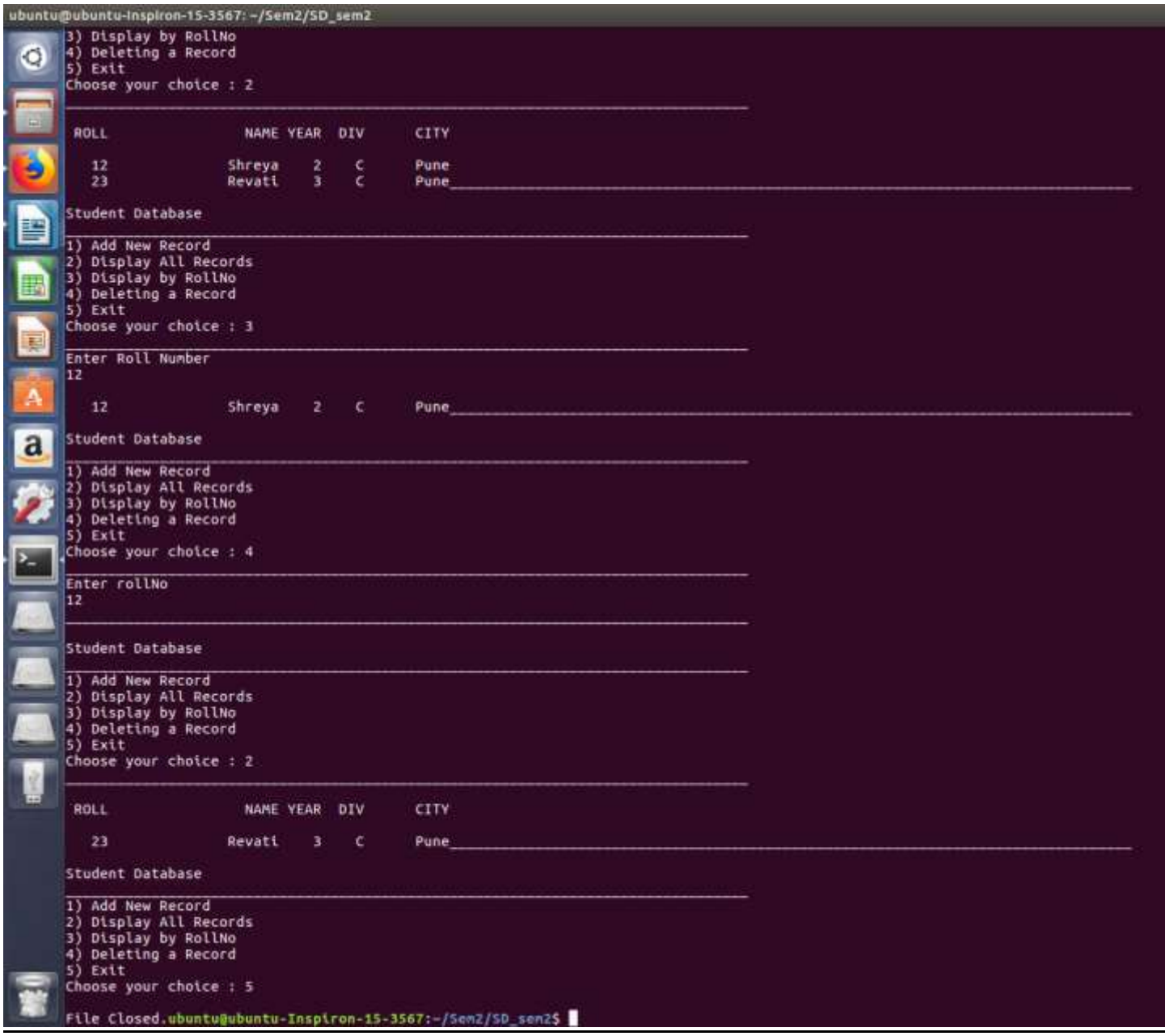
```
        while(file.read((char *)&s1, sizeof(Student)))
        {
         if(s1.getRollNo()==rollno)
         {
          flag=true;
          continue;
         }
         outFile.write((char *)&s1, sizeof(Student));
        }
        if(!flag)
        {
         cout<<"\nRecord of "<<rollno<<" is not present."<<endl;
        }
        file.close();
        outFile.close();
        remove("student.dat");
        rename("new.dat","student.dat");
        file.open("student.dat",ios::in|ios::out|ios::ate|ios::binary);
       }
       ~FileOperations()
       {
        file.close();
        cout<<"\nFile Closed.";
       }
};

int main() {
 ofstream newFile("student.dat",ios::app|ios::binary);
  newFile.close();
  FileOperations file((char*)"student.dat");
    int rollNo,year,choice=0;
    char div;
```

```
    char name[MAX],address[MAX];
    while(choice!=5)
    {
       //clrscr();

cout<<"_____
_____"<<endl;
       cout<<"\nStudent Database\n";

cout<<"_____
_____"<<endl;
       cout<<"1) Add New Record\n";
       cout<<"2) Display All Records\n";
       cout<<"3) Display by RollNo\n";
       cout<<"4) Deleting a Record\n";
       cout<<"5) Exit\n";
       cout<<"Choose your choice : ";
       cin>>choice;
       switch(choice)
       {
          case 1 : //New Record

cout<<"_____
_____"<<endl;
            cout<<endl<<"Enter RollNo and name : \n";
            cin>>rollNo>>name;
            cout<<"Enter Year and Division : \n";
            cin>>year>>div;
            cout<<"Enter address : \n";
            cin>>address;
            file.insertRecord(rollNo,name,year,div,address);
            cout<<"\nRecord Inserted."<<endl;
            break;
```

```
          case 2 :

cout<<"_____
_____"<<endl;

cout<<endl<<setw(5)<<"ROLL"<<setw(20)<<"NAME"<<setw(5)<<"YEAR"<
<setw(5)<<"DIV"<<setw(10)<<"CITY"<<endl;
            file.displayAll();
            break;
          case 3 :

cout<<"_____
_____"<<endl;
            cout<<"Enter Roll Number"<<endl;
            cin>>rollNo;
             file.displayRecord(rollNo);

            break;
          case 4:

cout<<"_____
_____"<<endl;
            cout<<"Enter rollNo"<<endl;
            cin>>rollNo;
            file.deleteRecord(rollNo);
            break;
         case 5 :break;
       }

    }

 return 0;
}
```

Output :


ubuntu@ubuntu-inspiron-15-3567: ~/Sem2/SD_sem2
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2/SD_sem2$ g++ sd8.cpp
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2/SD_sem2$ ./a.out
Student Database
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Deleting a Record
5) Exit
Choose your choice : 1
Enter RollNo and name :
12
Shreya
Enter Year and Division :
2
C
Enter address :
Pune
Record Inserted.
Student Database
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Deleting a Record
5) Exit
Choose your choice : 1
Enter RollNo and name :
23
Revati
Enter Year and Division :
3
C
Enter address :
Pune
Record Inserted.
Student Database
1) Add New Record
2) Display All Records
3) Display by RollNo
4) Deleting a Record
5) Exit
Choose your choice : 2
ROLL NAME YEAR DIV CITY
12 Shreya 2 C Pune
23 Revati 3 C Pune



Conclusion:- We have successfully implemented file handling and performed functions like insertion,deletion and display of record using sequential file.

Assignment Number 9

Aim:
Company maintains employee information as employee ID, name, designation and salary. Allow user to add, delete information of employee. Display information of particular employee. If employee does not exist an appropriate message is displayed. If it is, then the system displays the employee details. Use index sequential file to maintain the data.

Objective:
To study use of different data structure concepts in this program.

Theory:
Input/output formatting

Writing to or reading from a file is similar to writing onto a terminal screen or reading from a keyboard. Differences are:

- File must be opened with an OPEN statement, in which the unit number and (optionally) the filename are given
- Subsequent writes (or reads) must refer to a known unit number (used for open)
- File should be closed at the end

File opening and closing

The syntax is:

```
OPEN([unit=]lunit,file='name' [,options])

CLOSE([unit=]lunit [,options])
```

For example:

```
OPEN(10, file='output.dat', status='new')

CLOSE(unit=10)
```

- The first parameter is the unit number and the keyword unit= can be omitted.
- The unit numbers 0,5 and 6 are predefined.
  - 0 is output for standard (system) error messages
  - 5 is for standard (user) input
  - 6 is for standard (user) output
  - These units are opened by default and should not be re-opened nor closed by users

Some options for opening a file:

- status: existence of the file ('old', 'new', 'replace', 'scratch', 'unknown')
- position: offset, where to start writing ('append')
- action: file operation mode ('write','read','readwrite')
- form: text or binary file ('formatted', 'unformatted')
- access: direct or sequential file access ('direct','sequential','stream')
- iostat: error indicator, (output) integer (non zero only upon an error)
- err: the label number to jump upon error
- recl: record length, (input) integer for direct access files only. Be careful, it can be in bytes or words...

Program Code:

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

```
using namespace std;

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

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

//Display employee record
```

```
void Employee::display()
{
   cout<<code<<" "<<name<<"\t"<<salary<<endl;
}

//global declaration
fstream file;

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

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

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

   file.open("EMPLOYEE.DAT",ios::binary|ios::app);
   if(!file){

cout<<"_____
_____"<<endl;
      cout<<"ERROR IN CREATING FILE\n";
      return;
   }
   //write into file
   file.write((char*)&x,sizeof(x));
   file.close();
```

```
cout<<"_____
_____"<<endl;
   cout<<"Record added sucessfully.\n";
}

void displayAll(){
   Employee    x;

   file.open("EMPLOYEE.DAT",ios::binary|ios::in);
   if(!file){

cout<<"_____
_____"<<endl;
      cout<<"ERROR IN OPENING FILE \n";
      return;
   }
   while(file){
   if(file.read((char*)&x,sizeof(x)))
      if(x.getSalary()>=10000 && x.getSalary()<=20000)
         x.display();
   }
  file.close();
}

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

        cout<<"_____
_____"<<endl;
```

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


   file.open("EMPLOYEE.DAT",ios::binary|ios::in);
   if(!file){

cout<<"_____
_____"<<endl;
      cout<<"ERROR IN OPENING FILE \n";
      return;
   }
   while(file){
      if(file.read((char*)&x,sizeof(x))){
         if(x.getEmpCode()==c){

cout<<"_____
_____"<<endl;
            cout<<"RECORD FOUND\n";
            x.display();
            isFound=1;
            break;
         }
      }
   }
   if(isFound==0){

cout<<"_____
_____"<<endl;
      cout<<"Record not found!!!\n";
   }
   file.close();
}
```

```
//Function to increase salary
void increaseSalary(){
   //read employee id
   Employee    x;
   int c;
   int isFound=0;
   float sal;
       cout<<"_____
_____"<<endl;
   cout<<"enter employee code \n";
   cin>>c;


   file.open("EMPLOYEE.DAT",ios::binary|ios::in);
   if(!file){

cout<<"_____
_____"<<endl;
      cout<<"ERROR IN OPENING FILE \n";
      return;
   }
   while(file){
      if(file.read((char*)&x,sizeof(x))){
         if(x.getEmpCode()==c){
            cout<<"Salary hike? ";
            cin>>sal;
            x.updateSalary(x.getSalary()+sal);
            isFound=1;
            break;
         }
      }
   }
```

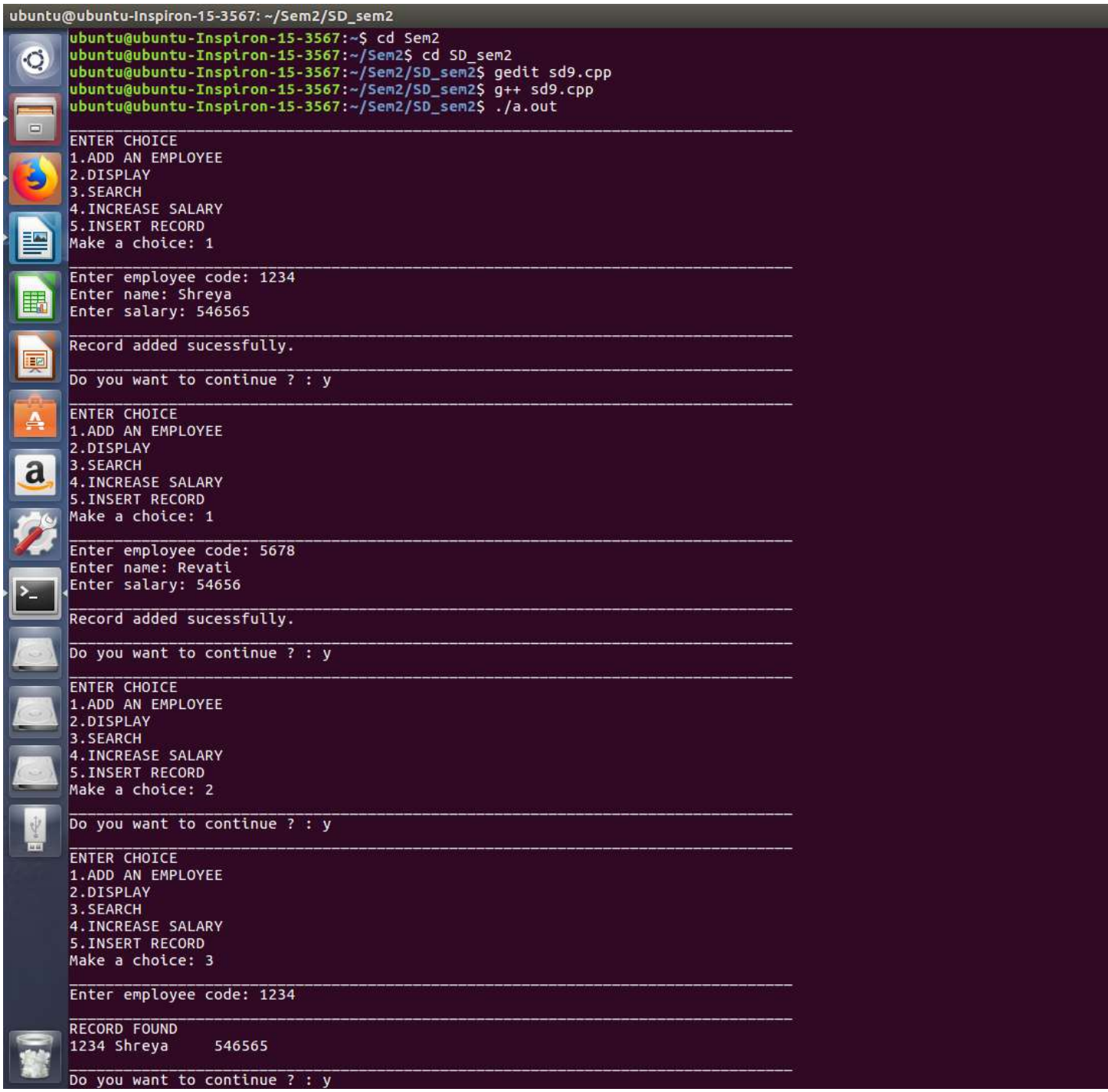
```
   if(isFound==0){

cout<<"_____
_____"<<endl;
      cout<<"Record not found!!!\n";
   }
   file.close();

cout<<"_____
_____"<<endl;
   cout<<"Salary updated successfully."<<endl;
}

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

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

   fstream fin;
   //read file in input mode
   file.open("EMPLOYEE.DAT",ios::binary|ios::in);
   //open file in write mode
   fin.open("TEMP.DAT",ios::binary|ios::out);

   if(!file){
      cout<<"Error in opening EMPLOYEE.DAT file!!!\n";
      return;
   }
```

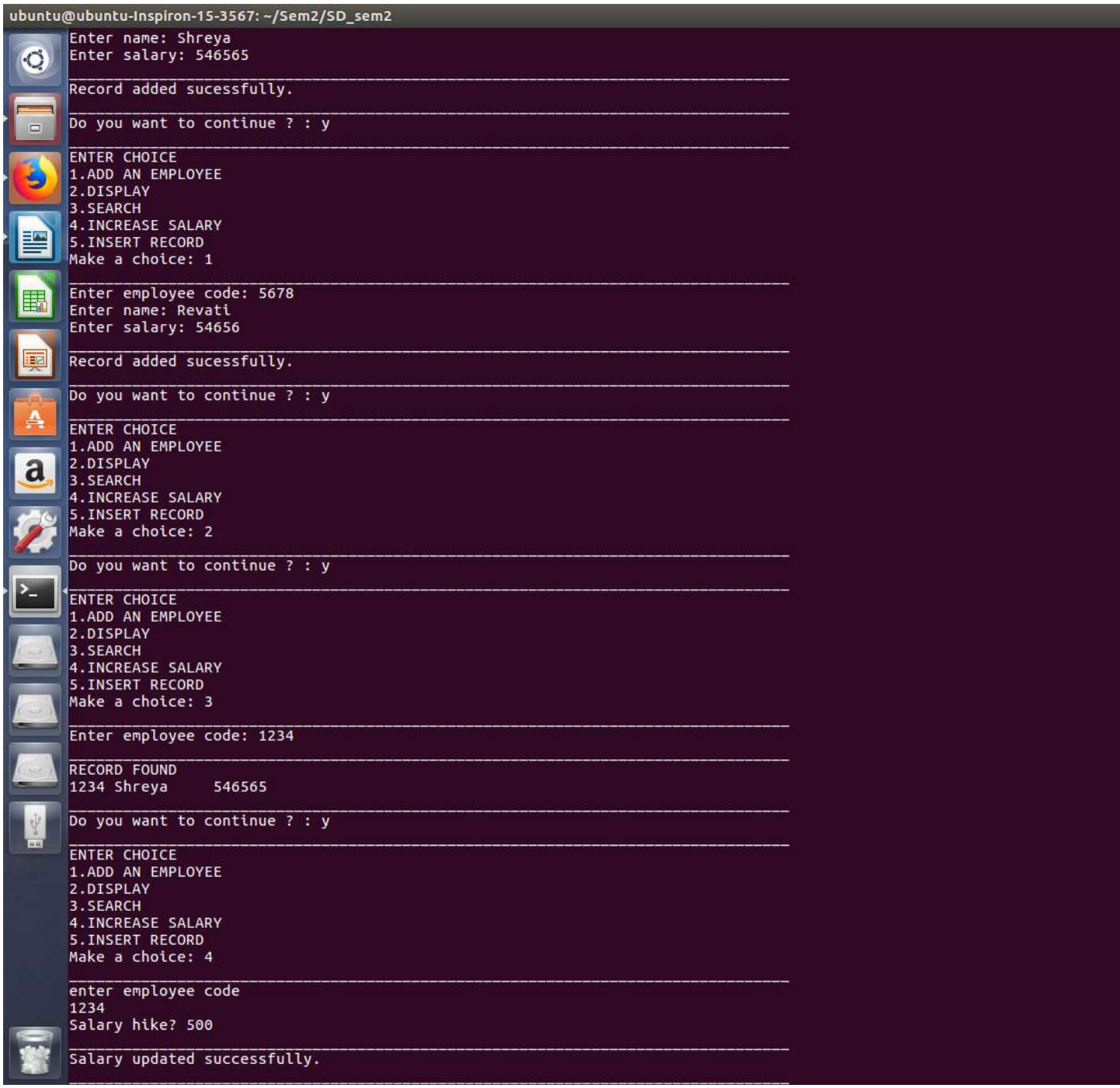
```
   if(!fin){
      cout<<"Error in opening TEMP.DAT file!!!\n";
      return;
   }
   while(file){
      if(file.read((char*)&x,sizeof(x))){
         if(x.getEmpCode()>newEmp.getEmpCode()){
            fin.write((char*)&newEmp, sizeof(newEmp));
         }
         //no need to use else
         fin.write((char*)&x,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

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

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

int main()
{
    char ch;

   //if required then only remove the file
    deleteExistingFile();

    do{
    int n;
        cout<<"_____
_____"<<endl;
```

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

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

        default :
            cout<<"Invalid Choice\n";
    }
        cout<<"_____
_____"<<endl;
    cout<<"Do you want to continue ? : ";
    cin>>ch;

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

```
    return 0;
}
```

Output:


ubuntu@ubuntu-Inspiron-15-3567: ~/Sem2/SD_sem2
ubuntu@ubuntu-Inspiron-15-3567:~$ cd Sem2
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2$ cd SD_sem2
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2/SD_sem2$ gedit sd9.cpp
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2/SD_sem2$ g++ sd9.cpp
ubuntu@ubuntu-Inspiron-15-3567:~/Sem2/SD_sem2$ ./a.out
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 1
Enter employee code: 1234
Enter name: Shreya
Enter salary: 546565
Record added sucessfully.
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 1
Enter employee code: 5678
Enter name: Revati
Enter salary: 54656
Record added sucessfully.
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 2
Do you want to continue ? : y
ENTER CHOICE
1.ADD AN EMPLOYEE
2.DISPLAY
3.SEARCH
4.INCREASE SALARY
5.INSERT RECORD
Make a choice: 3
Enter employee code: 1234
RECORD FOUND
1234 Shreya 546565
Do you want to continue ? : y



Conclusion: Thus, this assignment is completed successfully.