SEMESTER II

SKILL DEVELOPMENT

ASSIGNMENT 1

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

CODE:

```
#include<string.h>
#include <iostream>
using namespace std;
#define size 20

int linears(int ,int);
void create(int *s1,int *s2)
{
   int n,n2;
   cout<<"\nEnter number of elements for Set 1:";
   cin>>n;
   cout<<"\nEnter "<<n<< " elements:";
   s1[0]=n;
   for(int i=1;i<=n;i++)
   {
      cin>>s1[i];
   }
   cout<<"\nEnter number of elements for Set 2:";
   cin>>n2;
   cout<<"\nEnter "<<n2<< " elements:";
   s2[0]=n2;
   for(int i=1;i<=n2;i++)
   {
      cin>>s2[i];
   }
}

void display(int *s1,int *s2)
```

```
{
   cout<<"\nSet 1:";
   cout<<"\nNumber of elements in Set 1:"<<s1[0];
   for(int i=1;i<=s1[0];i++)
   {
      cout<<"\t"<<s1[i];
   }
   cout<<"\nSet 2:";
   cout<<"\nNumber of elements in Set 2:"<<s2[0];
   for(int i=1;i<=s2[0];i++)
   {
      cout<<"\t"<<s2[i];
   }

}

int linears(int *s,int e)
{
   for (int i=1;i<=s[0];i++)
      {
         if(s[i]==e)
         {
            return 1;
         }
      }
   return 0;
}


int subset(int *s1,int *s2){
   int i=1,status=1;
   while(i<=s2[0]){
      if(!linears(s1, s2[i])){
         status=0;
         break;
      }
      else{
         status=1;
      }
      i++;
   }
   return status;
```

```
}

void intersection(int *s1,int*s2,int *s3)
{
   int i=1,j=1;
   cout<<"\nIntersection of two sets:";
   while(i<=s2[0])
   {
      if(linears(s1, s2[i]))
      {
         s3[j]=s2[i];
         j++;
      }
      i++;
   }
   s3[0]=j;
   for(int j=1;j<s3[0];j++)
   {
      cout<<" "<<s3[j];
   }
}

void diffab(int *s1,int *s2,int *s3)
{
   int i=1,j=1;
   cout<<"Difference of two sets:(A-B):";
   while(i<=s1[0])
   {
      if(!linears(s2, s1[i]))
      {
         s3[j]=s1[i];
         j++;
      }
      i++;
   }
   s3[0]=j;
   for(int j=1;j<s3[0];j++)
   {
      cout<<" "<<s3[j];
   }
}
```

```
void diffba(int *s1,int *s2,int *s3)
{
   int i=1,j=1;
   cout<<"Difference of two sets:(B-A):";
   while(i<=s2[0])
   {
      if(!linears(s1, s2[i]))
      {
         s3[j]=s2[i];
         j++;
      }
      i++;
   }
   s3[0]=j;
   for(int j=1;j<s3[0];j++)
   {
      cout<<" "<<s3[j];
   }
}

void union1(int *s1,int *s2,int *s3)
{
   int i;
   cout<<"\nUnion of two sets";
   for(i=1;i<=s1[0];i++)
   {
      s3[i]=s1[i];
   }

   for(int j=1;j<=s2[0];j++)
      if(!linears(s1,s2[j]))
      {
         s3[i]=s2[j];
         i++;
      }

   s3[0]=i;
      for(int j=1;j<s3[0];j++)
      {
         cout<<" "<<s3[j];
      }
}
```

```
void insert(int *s,int e){
   s[0]++;
   s[s[0]]=e;
   cout<<"\nSet A:";
   for(int i=1;i<=s[0];i++){
      cout<<s[i]<<" ";
   }
}

void remove(int *s,int e){
   int i=1;
   while(i<=s[0]){
      if(s[i]==e){
         for(int j=i;j<s[0];j++){
            s[j]=s[j+1];
         }
         cout<<"\nElement deleted..";
         s[0]--;
         for(int i=1;i<= s[0];i++){
            cout<<s[i]<<" ";
         }
      }
      i++;
   }
   cout<<"\nElement not found!";
}

void contains(int *s,int e ){
   int i=1;
   while(i<=s[0]){
      if(s[i]==e){
         cout<<"\nElement found at position "<<i;
         break;
      }
      else{
         cout<<"\nElement not found.";
         break;
      }
   }
}
```

```
void size1(int *s){
   int count;
   count=s[0];
   cout<<count;
}


int main(int argc, const char * argv[]) {
   int s1[size],s2[size],s3[size];
   int ch,e,status;
   char set_ch;
   do{
      cout<<"\n---MENU---";
      cout<<"\n1.Create Set\n2.Display Set\n3.Union of
Sets\n4.Intersection of Sets\n5.Difference(A-B)\n6.Difference(B-
A)\n7.Subset\n8.Insert.\n9.Remove.\n10.Contains.\n11.Size";
      cout<<"\nEnter your choice:";
      cin>>ch;
      switch (ch) {
         case 1:
            create(s1,s2);
            break;
         case 2:
            display(s1,s2);
            break;
         case 3:
            union1(s1,s2,s3);
            break;
         case 4:
            intersection(s1, s2, s3);
            break;
         case 5:
            diffab(s1, s2, s3);
            break;
         case 6:
            diffba(s1, s2, s3);
            break;
         case 7:
            status=subset(s1,s2);
           (status==0)?cout<<"\nB Subset of A: false" :cout<<"\nB
Subset of A: true";
            break;
```

```
        case 8:
           cout<<"Which set you want to enter the element in?(A/B): ";
           cin>>set_ch;
           cout<<"Enter element to be inserted: ";
           cin>>e;
           if(set_ch=='A'||set_ch=='a')
           {
              insert(s1,e);
           }
           else
           {
              insert(s2,e);
           }
           break;
        case 9:
           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')
           {
              remove(s1,e);
           }
           else
           {
              remove(s2,e);
           }
           break;
        case 10:
           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')
           {
              contains(s1,e);
           }
           else
           {
              contains(s2,e);
```

```
            }
            break;
         case 11:
            cout<<"\nSize of set A:";
            size1(s1);
            cout<<"\nSize of set B:";
            size1(s2);
            break;
         default:
            break;

      }
   }while(ch<=10);
   return 0;
}
```

/*OUTPUT

```
---MENU---
1.Create Set
2.Display Set
3.Union of Sets
4.Intersection of Sets
5.Difference(A-B)
6.Difference(B-A)
7.Subset
8.Insert.
9.Remove.
10.Contains.
11.Size
Enter your choice:1

Enter number of elements for Set 1:5

Enter 5 elements:1
2
3
4
5

Enter number of elements for Set 2:3

Enter 3 elements:4
```

7
2

---MENU---
1.Create Set
2.Display Set
3.Union of Sets
4.Intersection of Sets
5.Difference(A-B)
6.Difference(B-A)
7.Subset
8.Insert.
9.Remove.
10.Contains.
11.Size
Enter your choice:3

Union of two sets 1 2 3 4 5 7
---MENU---
1.Create Set
2.Display Set
3.Union of Sets
4.Intersection of Sets
5.Difference(A-B)
6.Difference(B-A)
7.Subset
8.Insert.
9.Remove.
10.Contains.
11.Size
Enter your choice:4

Intersection of two sets: 4 2
---MENU---
1.Create Set
2.Display Set
3.Union of Sets
4.Intersection of Sets
5.Difference(A-B)
6.Difference(B-A)
7.Subset
8.Insert.

9.Remove.
10.Contains.
11.Size
Enter your choice:5
Difference of two sets:(A-B): 1 3 5
---MENU---
1.Create Set
2.Display Set
3.Union of Sets
4.Intersection of Sets
5.Difference(A-B)
6.Difference(B-A)
7.Subset
8.Insert.
9.Remove.
10.Contains.
11.Size
Enter your choice:6
Difference of two sets:(B-A): 7
---MENU---
1.Create Set
2.Display Set
3.Union of Sets
4.Intersection of Sets
5.Difference(A-B)
6.Difference(B-A)
7.Subset
8.Insert.
9.Remove.
10.Contains.
11.Size
Enter your choice:7

B Subset of A: false
---MENU---
1.Create Set
2.Display Set
3.Union of Sets
4.Intersection of Sets
5.Difference(A-B)
6.Difference(B-A)
7.Subset

8.Insert.
9.Remove.
10.Contains.
11.Size
Enter your choice:8
Which set you want to enter the element in?(A/B): 9
Enter element to be inserted: 10

Set A:4 7 2 10
---MENU---
1.Create Set
2.Display Set
3.Union of Sets
4.Intersection of Sets
5.Difference(A-B)
6.Difference(B-A)
7.Subset
8.Insert.
9.Remove.
10.Contains.
11.Size
Enter your choice:11

Size of set A:5
Size of set B:4
*/

ASSIGNMENT 2

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

CODE:

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

typedef struct node {
   int data;
   node *left;
   node *right;
   int lth;
   int rth;
}node;

node *root = NULL; node *dummy = NULL;

void insert_into_tbt(node *root, int value) {
   node *newnode = new node;
   newnode->data = value;
   newnode->left = NULL;
   newnode->right = NULL;
   newnode->lth = newnode->rth = 1;
   node *visitedNode, *parent;
   visitedNode = root;
   while(visitedNode != dummy) {
      parent = visitedNode;
      if(newnode->data > visitedNode->data) {
         if(visitedNode->rth == 1) {
            break;
         } else {
            visitedNode = visitedNode->right;
         }
      } else {
         if(visitedNode->lth == 1) {
```

```
            break;
         } else {
            visitedNode = visitedNode->left;
         }
      }
   }
   if(newnode->data > parent->data) {
      newnode->right = parent->right;
      newnode->left = parent;
      parent->rth = 0;
      parent->right = newnode;
   } else {
      newnode->left = parent->left;
      newnode->right = parent;
      parent->lth = 0;
      parent->left = newnode;
   }
}

void create_bst(int size, int *arr) {
   for(int i=0;i<size;i++) {
      if(root==NULL) {
         node *newnode = new node;
         newnode->data = arr[i];
         newnode->lth = newnode->rth = 1;
         dummy = new node;
         dummy->data = -999;
         dummy->left = newnode;
         dummy->right = dummy;
         dummy->lth = dummy->rth = 0;
         newnode->left = newnode->right = dummy;
         root = newnode;
      } else {
         insert_into_tbt(root, arr[i]);
      }
   }
}

void display_tbt() {
   node *temp = root;
   while(temp != dummy) {
      while(temp->lth != 1) {
```

```
            temp = temp->left;
        }
        cout << temp->data << "\t";
        if(temp->rth == 0) {
            temp = temp->right;
        } else {
            while(temp->rth == 1) {
                temp = temp->right;

                if(temp == dummy) {
                    break;
                }
                cout << temp->data << "\t";
            }
            temp = temp->right;
        }
    }
    cout << endl;
}

int main() {
    int no_of_nodes, choice;
    int *nodeValues;

    do {
        cout << "\n1. CREATE\t2. DISPLAY\t0. EXIT\n";
        cout << "Enter your choice : ";
        cin >> choice;
        switch(choice) {
            case 1:
                cout << "\nEnter no. of nodes : ";
                cin >> no_of_nodes;
                nodeValues = new int[no_of_nodes];
                cout << "Enter Values : \n";
                for(int i=0;i<no_of_nodes;i++) {
                    cin >> nodeValues[i];
                }
                create_bst(no_of_nodes,nodeValues);
                break;
            case 2:
                cout << "\nINORDER : ";
                display_tbt();
```

```
            break;
        }

    }while(choice!=0);
    return 0;
}
/*OUTPUT
1. CREATE    2. DISPLAY    0. EXIT
 Enter your choice : 1

 Enter no. of nodes : 4
 Enter Values :
 10
 23
 1
 45

 1. CREATE    2. DISPLAY    0. EXIT
 Enter your choice : 2

 INORDER : 1    10    23    45
*/
```

ASSIGNMENT 3

AIM: There are flight paths between cities. If there is a flight between city A and city B then there is an edge between the cities. The cost of the edge can be the time that flight takes to reach city B from A, or the amount of fuel used for the journey. Represent this as a graph. The node can be represented by airport name or name of the city. Use adjacency list representation of the graph or use adjacency matrix representation of the graph. Justify the storage representations used.

CODE:

```
#include<iostream>
using namespace std;
#define INFINITY 9999
#define MAX 10

void dijkstra(int G[MAX][MAX],int n,int startnode);

int main()
{
   int G[MAX][MAX],i,j,n,u;
   cout<<"Enter no. of vertices:";
   cin>>n;
   cout<<"\nEnter the adjacency matrix:\n";

   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
         cin>>G[i][j];

   cout<<"\nEnter the starting node:";
   cin>>u;
   dijkstra(G,n,u);

   return 0;
}
```

```
void dijkstra(int G[MAX][MAX],int n,int startnode)
{

   int cost[MAX][MAX],distance[MAX],pred[MAX];
   int visited[MAX],count,mindistance,nextnode,i,j;

   //pred[] stores the predecessor of each node
   //count gives the number of nodes seen so far
   //create the cost matrix
   for(i=0;i<n;i++)
   {
      for(j=0;j<n;j++)
      {
         if(G[i][j]==0)
            cost[i][j]=INFINITY;
         else
            cost[i][j]=G[i][j];
      }
   }
   //initialize pred[],distance[] and visited[]
   for(i=0;i<n;i++)
   {
      distance[i]=cost[startnode][i];
      pred[i]=startnode;
      visited[i]=0;
   }

   distance[startnode]=0;
   visited[startnode]=1;
   count=1;

   while(count<n-1)
   {
      mindistance=INFINITY;

      //nextnode gives the node at minimum distance
      for(i=0;i<n;i++)
         if(distance[i]<mindistance&&!visited[i])
         {
            mindistance=distance[i];
            nextnode=i;
         }
```

```
    //check if a better path exists through nextnode
    visited[nextnode]=1;
    for(i=0;i<n;i++)
      if(!visited[i])
        if(mindistance+cost[nextnode][i]<distance[i])
        {
          distance[i]=mindistance+cost[nextnode][i];
          pred[i]=nextnode;
        }
    count++;
  }

  //print the path and distance of each node
  for(i=0;i<n;i++)
    if(i!=startnode)
    {
      cout<<"\nDistance of node:"<<i<<" "<<distance[i];
      cout<<"\nPath"<<i;

      j=i;
      do
      {
        j=pred[j];
        cout<<"<-"<<j;
      }while(j!=startnode);
    }
}
```

/***OUTPUT**

```
Enter no. of vertices:5

Enter the adjacency matrix:
0 10 0 30 100
10 0 50 0 0
0 50 0 20 10
30 0 20 0 60
100 0 10 60 0

Enter the starting node:0

Distance of node:1 10
Path1<-0
```

Distance of node:2 50
Path2<-3<-0
Distance of node:3 30
Path3<-0
Distance of node:4 60
Path4<-2<-3<-0
*/

ASSIGNMENT 4

AIM: For a weighted graph G, find the minimum spanning tree using Prims algorithm

CODE:

```
#include<iostream>
#include<stdlib.h>
using namespace std;
#define infinity 9999
#define MAX 20

int G[MAX][MAX],spanning[MAX][MAX],n;

int prims();

int main()
{
   int i,j,total_cost;
   printf("Enter no. of vertices:");
   scanf("%d",&n);

   printf("\nEnter the adjacency matrix:\n");

   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
         scanf("%d",&G[i][j]);

   total_cost=prims();
   printf("\nspanning tree matrix:\n");

   for(i=0;i<n;i++)
   {
      printf("\n");
      for(j=0;j<n;j++)
         printf("%d\t",spanning[i][j]);
   }
```

```
   printf("\n\nTotal cost of spanning tree=%d",total_cost);
   return 0;
}

int prims()
{
   int cost[MAX][MAX];
   int u,v,min_distance,distance[MAX],from[MAX];
   int visited[MAX],no_of_edges,i,min_cost,j;

   //create cost[][] matrix,spanning[][]
   for(i=0;i<n;i++)
      for(j=0;j<n;j++)
      {
         if(G[i][j]==0)
            cost[i][j]=infinity;
         else
            cost[i][j]=G[i][j];
         spanning[i][j]=0;
      }

   //initialise visited[],distance[] and from[]
   distance[0]=0;
   visited[0]=1;

   for(i=1;i<n;i++)
   {
      distance[i]=cost[0][i];
      from[i]=0;
      visited[i]=0;
   }

   min_cost=0;       //cost of spanning tree
   no_of_edges=n-1;       //no. of edges to be added

   while(no_of_edges>0)
   {
      //find the vertex at minimum distance from the tree
      min_distance=infinity;
      for(i=1;i<n;i++)
         if(visited[i]==0&&distance[i]<min_distance)
```

```
        {
          v=i;
          min_distance=distance[i];
        }

      u=from[v];

      //insert the edge in spanning tree
      spanning[u][v]=distance[v];
      spanning[v][u]=distance[v];
      no_of_edges--;
      visited[v]=1;

      //updated the distance[] array
      for(i=1;i<n;i++)
        if(visited[i]==0&&cost[i][v]<distance[i])
        {
          distance[i]=cost[i][v];
          from[i]=v;
        }

      min_cost=min_cost+cost[u][v];
    }

    return(min_cost);
}
/*OUTPUT
 Enter no. of vertices:5

 Enter the adjacency matrix:
 0 10 0 30 100
 10 0 50 0 0
 0 50 0 20 10
 30 0 20 0 60
 100 0 10 60 0

 spanning tree matrix:

 0   10   0   30   0
 10   0   0   0   0
 0   0   0   20   10
 30   0   20   0   0
```

```
0   0   10   0   0

Total cost of spanning tree=70
*/
```

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

CODE:

```
#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;        //stores all edges in graph and are sorted in
increasing order of their weight.

int G[MAX][MAX],n;
edgelist spanlist;         // list of edges in spanning tree

void kruskal();
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;
   elist.count=0;

   for(i=1;i<n;i++)
      for(j=0;j<i;j++)
      {
         if(G[i][j]!=0)
         {
            elist.data[elist.count].u=i;
            elist.data[elist.count].v=j;
            elist.data[elist.count].w=G[i][j];
            elist.count++;
         }
      }

   sort();

   for(i=0;i<n;i++)
      belongs[i]=i;

   spanlist.count=0;
```

```
   for(i=0;i<elist.count;i++)
   {
      cno1=find(belongs,elist.data[i].u);
      cno2=find(belongs,elist.data[i].v);

      if(cno1!=cno2)
      {
         spanlist.data[spanlist.count]=elist.data[i];
         spanlist.count=spanlist.count+1;
         union1(belongs,cno1,cno2);
      }
   }
}

int find(int belongs[],int vertexno)
{
   return(belongs[vertexno]);
}

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

   for(i=0;i<n;i++)
      if(belongs[i]==c2)
         belongs[i]=c1;
}

void sort()
{
   int i,j;
   edge temp;

   for(i=1;i<elist.count;i++)
      for(j=0;j<elist.count-1;j++)
         if(elist.data[j].w>elist.data[j+1].w)
         {
            temp=elist.data[j];
            elist.data[j]=elist.data[j+1];
            elist.data[j+1]=temp;
         }
}
```

```
void print()
{
   int i,cost=0;

   for(i=0;i<spanlist.count;i++)
   {
      cout<<"\n"<<spanlist.data[i].u<<"  "<<spanlist.data[i].v<<"
"<<spanlist.data[i].w;
      cost=cost+spanlist.data[i].w;
   }

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

/***OUTPUT**
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
5 2 4

Minimum cost of the telephone lines between the cities:13*/

ASSIGNMENT 6

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

CODE:

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

void heapify(int a[],int i,int n);
void heapsort(int a[],int n);
void print(int a[],int n);

int main()
{
   int a[max],n,i;
   cout<<"\nEnter the number of values:";
   cin>>n;
   cout<<"\nEnter "<<n<<" values:";
   for(i=0;i<n;i++)
   {
      cin>>a[i];
   }

   heapsort(a, n);

   cout<<"\nSorted array:";
   print(a,n);
}



void heapify(int a[],int i,int n)
{
   int heap_size=n;
```

```
    int largest=i,temp;
    int l=2*i+1;
    int r=2*i+2;
    if(l<heap_size && a[l]>a[largest])
    {
       largest=l;
    }

    if(r<heap_size && a[r]>a[largest])
    {
       largest=r;
    }

    if(largest!=i)
    {
       temp=a[i];
       a[i]=a[largest];
       a[largest]=temp;
       heapify(a,largest,n);
    }

}

 void heapsort(int a[],int n)
 {
    int temp;
    for(int i=n/1-2;i>=0;i--)
       heapify(a, i, n);
    for(int i=n-1;i>=0;i--)
    {
       temp=a[0];
       a[0]=a[i];
       a[i]=temp;
       heapify(a,0,i);
    }

 }

 void print(int a[],int n)
 {
    for(int i=0;i<n;i++)
    {
```

```
      cout<<a[i]<<" ";
   }
}

/*OUTPUT
 Enter the number of values:5

 Enter 5 values:23 54 1 45 3

 Sorted array:1 3 23 45 54
 */
```

ASSIGNMENT 7

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

CODE:

```
#include <iostream>
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)
      {
         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;
   }
}
```

```
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)
         return key;
      else
         return -1;
   }
}
int main()
{
   int value,s,n;
   char choice='y';
   cout<<" \nEnter the size of hash table "<<endl;
   cin>>n;
   int a[n];
   for (int i=0;i<n;i++)
      a[i]=-1;
   while (choice=='y')
   {
      cout<<endl<<"..... MENU ....."<<endl;
```

```
      cout<<"1.For read number of element and factor present in
hashtable "<<endl;
      cout<<"2. For to add value into the hashtable "<<endl;
      cout<<"3. To print the hashtable "<<endl;
      cout<<"4. To search the element "<<endl;
      cout<<"5. To exit "<<endl;
      cin>>s;
      switch(s)
      {
         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<<" E   nter 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;
            break;
         case 5:
            choice= 'n';
            break;
      }
   }

   return 0;
}
/*OUTPUT
Enter the size of hash table
```

```
5

..... MENU .....
1.For read number of element and factor present in hashtable
2. For to add value into the hashtable
3. To print the hashtable
4. To search the element
5. To exit
1
Total element in the set are 0
Load factor of hashtable is 0

..... MENU .....
1.For read number of element and factor present in hashtable
2. For to add value into the hashtable
3. To print the hashtable
4. To search the element
5. To exit
2

Enter the value to store in hashtable
12

..... MENU .....
1.For read number of element and factor present in hashtable
2. For to add value into the hashtable
3. To print the hashtable
4. To search the element
5. To exit
3
Index   Value
2        12

..... MENU .....
1.For read number of element and factor present in hashtable
2. For to add value into the hashtable
3. To print the hashtable
4. To search the element
5. To exit

*/
```

ASSIGNMENT 8

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

CODE:

```
#include <iostream>
#include<fstream>
#include<string>
using namespace std;
class student
{
  int id;
  string name,div,addr;
public:
  void set_info()
  {
    cout<<"Enter roll no.: ";
    cin>>id;
    cout<<"Enter name: ";
    cin>>name;
    cout<<"Enter division: ";
    cin>>div;
    cout<<"Enter address: ";
    cin>>addr;
  }
  void show_info()
  {
    cout<<"The name is "<<name<<" and the roll no is "<<id<<endl;
  }
  void write_record(student obj)
  {
```

```
        ofstream fout;
        fout.open("hii.txt",ios::binary|ios::app);
        fout.write((char*)&obj,sizeof(obj));
        fout.close();
        cout<<"Data written\n";
    }
    void read_record(int id)
    {
        ifstream fin;
        student obj;
        int loc,i=0;
        fin.open("hii.txt",ios::binary);
        fin.seekg(0, ios::beg);
        while(fin.read((char*)&obj,sizeof(obj)))
        {
            loc=sizeof(obj)*i;
            fin.seekg(loc, ios::beg);
            fin.read((char*)&obj,sizeof(obj));
            obj.show_info();
            if(obj.id==id)
            {
                obj.show_info();
                cout<<"This\n";
                break;
            }
            i++;
        }
        if(obj.id!=id)
        {
            cout<<"Error 404! Record not found\n";
        }
    }
};
int main()
{
    char op;
    do
    {
        int c;

cout<<"======================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: {
              student s;
              s.set_info();
              s.write_record(s);
           }
              break;
           case 2: {
              int id;
              student s;
              cout<<"Enter roll no. of record you wish to search\n";
              cin>>id;
              s.read_record(id);
           }
              break;
           default:cout<<"Invalid\n";
        }
        cout<<"Do you wish to go again?(y/n): ";
        cin>>op;
     }while(op=='y' || op=='Y');
     return 0;
}
```

/***OUTPUT**

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

_____
Enter your choice: 1
Enter roll no.: 1
Enter name: Pooja
Enter division: A
Enter address: KOndhw
Data written
Do you wish to go again?(y/n): Y
======================Menu====================
1] Add record
```

2] Search record
3] Delete record

_____

Enter your choice: 1
Enter roll no.: 10
Enter name: Ruchota
Enter division: B
Enter address: Pune
Data written
Do you wish to go again?(y/n): y
==========Menu==========
1] Add record
2] Search record
3] Delete record

_____

Enter your choice: 2
Enter roll no. of record you wish to search
10
The name is Ruchota and the roll no is 10
*/

ASSIGNMENT 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.

CODE:

```
#include<iostream>
#include<iomanip>
#include<fstream.h>
#include<string.h>
#include<stdlib.h>
class EMP_CLASS
{
   typedef struct EMPLOYEE
   {
      char name[10];
      int emp_id;
      int salary;
   }Rec;
   typedef struct INDEX
   {
      int emp_id;
      int position;
   }Ind_Rec;
   Rec Records;
   Ind_Rec Ind_Records;
public:
   EMP_CLASS();
   void Create();
   void Display();
   void Update();
   void Delete();
```

```
   void Append();
   void Search();
};
EMP_CLASS::EMP_CLASS()//constructor
{
   strcpy(Records.name,"");
}
void EMP_CLASS::Create()
{
   int i,j;
   char ch='y';
   fstream seqfile;
   fstream indexfile;
   i=0;
   indexfile.open("IND.DAT",ios::in|ios::out|ios::binary);
   seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
   do
   {
      cout<<"\n Enter Name: ";
      cin>>Records.name;
      cout<<"\n Enter Emp_ID: ";
      cin>>Records.emp_id;
      cout<<"\n Enter Salary: ";
      cin>>Records.salary;
      seqfile.write((char*)&Records,sizeof(Records))<<flush;
      Ind_Records.emp_id=Records.emp_id;
      Ind_Records.position=i;

indexfile.write((char*)&Ind_Records,sizeof(Ind_Records))<<flush;
      i++;
      cout<<"\nDo you want to add more records?";
      cin>>ch;
   }while(ch=='y');
   seqfile.close();
   indexfile.close();
}
void EMP_CLASS::Display()
{
   fstream seqfile;
   fstream indexfile;
   int n,i,j;
   seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
```

```
   indexfile.open("IND.DAT",ios::in|ios::out|ios::binary);
   indexfile.seekg(0,ios::beg);
   seqfile.seekg(0,ios::beg);
   cout<<"\n The Contents of file are ..."<<endl;
   i=0;
   while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
   {

      i=Ind_Records.position*sizeof(Rec);//getting pos from index file
      seqfile.seekg(i,ios::beg);//seeking record of that pos from seq.file
      seqfile.read((char *)&Records,sizeof(Records));//reading record
      if(Records.emp_id!=-1)//if rec. is not deleted logically
      {   //then display it
         cout<<"\nName: "<<Records.name<<flush;
         cout<<"\nEmp_ID: "<<Records.emp_id;
         cout<<"\nSalary: "<<Records.salary;
         cout<<"\n";
      }

   }
   seqfile.close();
   indexfile.close();
}
void EMP_CLASS::Update()
{
   int pos,id;
   char New_name[10];
   int New_emp_id;
   int New_salary;
   cout<<"\n For updation,";
   cout<<"\n Enter the Emp_ID for for searching ";
   cin>>id;
   fstream seqfile;
   fstream indexfile;
   seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
   indexfile.open("IND.DAT",ios::in|ios::out|ios::binary);
   indexfile.seekg(0,ios::beg);

   pos=-1;
   //reading index file for getting the index
   while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
   {
```

```
      if(id==Ind_Records.emp_id)//the desired record is found
      {
         pos=Ind_Records.position;//getting the position
         break;
      }
   }
   if(pos==-1)
   {
      cout<<"\n The record is not present in the file";
      return;
   }
   else
   {
      cout<<"\n Enter the values for updation...";
      cout<<"\n Name: ";cin>>New_name;
      cout<<"\n Salary: ";cin>>New_salary;
      //calculating the position of record in seq. file using the pos of
ind. file
      int offset=pos*sizeof(Rec);
      seqfile.seekp(offset);//seeking the desired record for
modification
      strcpy(Records.name,New_name);//can be updated
      Records.emp_id=id;//It's unique id,so don't change
      Records.salary=New_salary;//can be updated
      seqfile.write((char*)&Records,sizeof(Records))<<flush;
      cout<<"\n The record is updated!!!";
   }
   seqfile.close();
   indexfile.close();

}
void EMP_CLASS::Delete()
{
   int id,pos;
   cout<<"\n For deletion,";
   cout<<"\n Enter the Emp_ID for for searching ";
   cin>>id;
   fstream seqfile;
   fstream indexfile;
   seqfile.open("EMP.DAT",ios::in|ios::out|ios::binary);
   indexfile.open("IND.DAT",ios::in|ios::out|ios::binary);
   seqfile.seekg(0,ios::beg);
```

```
   indexfile.seekg(0,ios::beg);
   pos=-1;
   //reading index file for getting the index
   while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
   {
      if(id==Ind_Records.emp_id) //desired record is found
      {
         pos=Ind_Records.position;
         Ind_Records.emp_id=-1;
         break;
      }
   }
   if(pos==-1)
   {
      cout<<"\n The record is not present in the file";
      return;
   }
   //calculating the position of record in seq. file using the pos of ind.
file
   int offset=pos*sizeof(Rec);
   seqfile.seekp(offset);//seeking the desired record for deletion
   strcpy(Records.name,"");
   Records.emp_id=-1; //logical deletion
   Records.salary=-1; //logical deletion
   seqfile.write((char*)&Records,sizeof(Records))<<flush;//writing
deleted status
   //From index file also the desired record gets deleted as follows
   offset=pos*sizeof(Ind_Rec);//getting position in index file
   indexfile.seekp(offset); //seeking that record
   Ind_Records.emp_id=-1; //logical deletion of emp_id
   Ind_Records.position=pos;//position remain unchanged
   indexfile.write((char*)&Ind_Records,sizeof(Ind_Records))<<flush;
   seqfile.seekg(0);
   indexfile.close();
   seqfile.close();
   cout<<"\n The record is Deleted!!!";
}
void EMP_CLASS::Append()
{
   fstream seqfile;
   fstream indexfile;
   int pos;
```

```
   indexfile.open("IND.DAT",ios::in|ios::binary);
   indexfile.seekg(0,ios::end);
   pos=indexfile.tellg()/sizeof(Ind_Records);
   indexfile.close();

   indexfile.open("IND.DAT",ios::app|ios::binary);
   seqfile.open("EMP.DAT",ios::app|ios::binary);

   cout<<"\n Enter the record for appending";
   cout<<"\nName: ";
   cin>>Records.name;
   cout<<"\nEmp_ID: ";
   cin>>Records.emp_id;
   cout<<"\nSalary: ";
   cin>>Records.salary;
   seqfile.write((char*)&Records,sizeof(Records)); //inserting rec at
end in seq. file
   Ind_Records.emp_id=Records.emp_id;          //inserting rec at end
in ind. file
   Ind_Records.position=pos;                   //at calculated pos
   indexfile.write((char*)&Ind_Records,sizeof(Ind_Records))<<flush;
   seqfile.close();
   indexfile.close();
   cout<<"\n The record is Appended!!!";
}
void EMP_CLASS::Search()
{
   fstream seqfile;
   fstream indexfile;
   int id,pos,offset;
   cout<<"\n Enter the Emp_ID for searching the record ";
   cin>>id;
   indexfile.open("IND.DAT",ios::in|ios::binary);
   pos=-1;
   //reading index file to obtain the index of desired record
   while(indexfile.read((char *)&Ind_Records,sizeof(Ind_Records)))
   {
      if(id==Ind_Records.emp_id)//desired record found
      {
         pos=Ind_Records.position;//seeking the position
         break;
      }
```

```
    }
    if(pos==-1)
    {
       cout<<"\n Record is not present in the file";
       return;
    }
    //calculate offset using position obtained from ind. file
    offset=pos*sizeof(Records);
    seqfile.open("EMP.DAT",ios::in|ios::binary);
    //seeking the record from seq. file using calculated offset
    seqfile.seekg(offset,ios::beg);//seeking for reading purpose
    seqfile.read((char *)&Records,sizeof(Records));
    if(Records.emp_id==-1)
    {
       cout<<"\n Record is not present in the file";
       return;
    }
    else //emp_id=desired record's id
    {
       cout<<"\n The Record is present in the file and it is...";
       cout<<"\n Name: "<<Records.name;
       cout<<"\n Emp_ID: "<<Records.emp_id;
       cout<<"\n Salary: "<<Records.salary;
    }
    seqfile.close();
    indexfile.close();
}
void main()
{
    EMP_CLASS List;
    char ans='y';
    int choice,key;
    clrscr();
    do
    {
       cout<<"\n          Main Menu          "<<endl;
       cout<<"\n 1.Create";
       cout<<"\n 2.Display";
       cout<<"\n 3.Update";
       cout<<"\n 4.Delete";
       cout<<"\n 5.Append";
       cout<<"\n 6.Search";
```

```
        cout<<"\n 7.Exit";
        cout<<"\n Enter your choice: ";
        cin>>choice;
        switch(choice)
        {
           case 1:List.Create();
              break;
           case 2:List.Display();
              break;
           case 3:List.Update();
              break;
           case 4:List.Delete();
              break;
           case 5:List.Append();
              break;
           case 6:List.Search();
              break;
           case 7:exit(0);
        }
        cout<<"\n\t Do you want to go back to Main Menu?";
        cin>>ans;
     }while(ans=='y');
}
```