/\*Problem Statement-: To create ADT that implements the SET concept.

a)Add (newElement) -Place a value into the set

b)Remove (element) Remove the value

c) Contains (element) Return true if element is in collection

d)Size () Return number of values in collection Iterator () Return an iterator used to loop

over collection

e)Intersection of two sets,

f)Union of two sets,

g) Difference between two sets,

h) Subset\*/

#include<iostream>

#include<cstdlib>

using namespace std;

template<class T>

class set

{

public:

T set1[50];

T set2[50];

int n1,n2;

set()

{

for(int i=0;i<50;i++)

{

set1[i]=-1;

set2[i]=-1;

}

n1=0;

n2=0;

}

int search(int , T);

void delet( );

void add();

void inter();

void unin();

void diff(int , int);

};

template<class T>

void set<T>::add( )

{

T data;

int s=0,i=-1;

cout<<"\nElement to be added in set-1 or set-2?(1/2)";

cin>>s;

cout<<"\nEnter element to be added-: ";

cin>>data;

if(s==1)

{

i=search(1,data);

if(i==-1)

{

set1[n1]=data;

n1++;

}

else

{

cout<<"\nElement already present!!";

}

}

else if(s==2)

{

i=search(2,data);

if(i==-1)

{

set2[n2]=data;

n2++;

}

else

{

cout<<"\nElement already present!!";

}

}

else

{

cout<<"\nEnter valid choice!!";

}

}

template<class T>

void set<T>::delet()

{

T data;

int s=0;

cout<<"\nElement to be deleted from set-1 or set-2?(1/2)";

cin>>s;

cout<<"\nEnter the element to be deleted-: ";

cin>>data;

int i=0;

if(s==1)

{

i=search(1,data);

if(i==-1)

{

cout<<"\nElement not found!!";

}

else

{

for(int j=i;j<n1-1;j++)

{

set1[j]=set1[j+1];

}

n1--;

}

}

else if(s==2)

{

i=search(2,data);

if(i==-1)

{

cout<<"\nElement not found!!";

}

else

{

for(int j=i;j<n2-1;j++)

{

set2[j]=set2[j+1];

}

n2--;

}

}

else

{

cout<<"\nEnter valid choice!!";

}

}

template<class T>

int set<T>::search(int no, T data)

{

int flag=-1;

if(no==1)

{

for(int i=0;i<n1;i++)

{

if(set1[i]==data)

{

flag=i;

break;

}

}

}

else if(no==2)

{

for(int i=0;i<n2;i++)

{

if(set2[i]==data)

{

flag=i;

break;

}

}

}

else

{

cout<<"\nEnter valid set number!!";

}

return flag;

}

template<class T>

void set<T>::inter()

{

T inte[50];

int k=0,i=0;

while(i<n1)

{

for(int j=0;j<n2;j++)

{

if(set1[i]==set2[j])

{

inte[k]=set2[j];

k++;

goto label;

}

}

label:i++;

}

cout<<"\nIntersection of two sets is-: \n";

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

{

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

}

}

template<class T>

void set<T>::unin()

{

T un[50];

int i=0,k=0;

for(int i=0;i<n2;i++)

{

un[k]=set2[i];

k++;

}

i=0;

while(i<n1)

{

for(int j=0;j<n2;j++)

{

if(set1[i]==set2[j])

{

goto label;

}

}

un[k]=set1[i];

k++;

label:i++;

}

cout<<"\nUnion of two sets is-: \n";

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

{

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

}

}

template<class T>

void set<T>::diff(int s1, int s2)

{

int i=0,k=0;

T dif[50];

if(s1==1 && s2==2)

{

while(i<n1)

{

for(int j=0;j<n2;j++)

{

if(set1[i]==set2[j])

{

goto label;

}

}

dif[k]=set1[i];

k++;

label:i++;

}

}

else if(s1==2 && s2==1)

{

while(i<n2)

{

for(int j=0;j<n1;j++)

{

if(set1[j]==set2[i])

{

goto label1;

}

}

dif[k]=set2[i];

k++;

label1:i++;

}

}

cout<<"\nDifference between set-"<<s1<<" and set-"<<s2<<"is-: \n";

{

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

{

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

}

}

}

int main()

{

set<int> s;

int ch,n,pos=-1,type;

int dat;

while(1)

{

cout<<"\n1.Add Element \n2.Delete Element \n3.Search Element \n4.Find Intersection \n5.Find Union \n6.Find Difference \n7.Exit";

cout<<"\nEnter choice-: ";

cin>>ch;

switch(ch)

{

case 1:

s.add();

break;

case 2:

s.delet();

break;

case 3:

cout<<"\nEnter set in which data is to be searched-: ";

cin>>n;

cout<<"\nEnter data to be searched-: ";

cin>>dat;

pos=s.search(n,dat);

if(pos==-1)

{

cout<<"\nData not found!!";

}

else

{

cout<<"\nData found at position -:"<<pos+1;

}

break;

case 4:

s.inter();

break;

case 5:

s.unin();

break;

case 6:

cout<<"\n1.A-B \n2.B-A\n";

cin>>n;

if(n==1)

{

s.diff(1,2);

}

else

{

s.diff(2,1);

}

break;

case 7:

exit(1);

default:

cout<<"\nEnter valid choice!!";

break;

}

}

}