```
//Attached: HW_10a
//=================================================
//Program: HW_10a
//=================================================
//Programmer: Ethan Nguyen
//Class: CS CMPR 131
//=================================================

#include "IntegerSet.h"

int main()
{
   const int SIZE = 4;
   int arr[SIZE] = {3, 76, 34, 56};

   IntegerSet s1;
   IntegerSet s2(arr, SIZE);
   IntegerSet s3;
   IntegerSet s4;

   cout << "Default set (s1):  { ";
   s1.printSet();
   cout << "}.";

   cout << "\n\nInserting 50, 100, 1, 0, 34, 56 to s1";
   s1.insertElement(50);
   s1.insertElement(100);
   s1.insertElement(1);
   s1.insertElement(0);
   s1.insertElement(34);
   s1.insertElement(56);

   cout << "\n\nNow, S1 = { ";
   s1.printSet();
   cout << "}.\n\n";

   cout << "Using array as parameter, set s2 = { ";
   s2.printSet();
   cout << "}.\n\n";

   cout << "Union of s1 and s2, set s3 = { ";
   s3 = s1.unionOfSets(s2);
   s3.printSet();
   cout << "}.\n\n";
```

```
    cout << "Intersection of s1 and s2, set s3 = { ";
    s3 = s1.intersectionOfSets(s2);
    s3.printSet();
    cout << "}.\n\n";

    cout << "Inserting 2 elements (34 and 56) into s4...\n";
    s4.insertElement(34);
    s4.insertElement(56);
    cout << "set s4 = { ";
    s4.printSet();
    cout << "}.\n\n";

    cout << "s2 == s3 = " << (s2.isEqualTo(s3) ? "True" : "False") << "\n\n";
    cout << "s3 == s4 = " << (s3.isEqualTo(s4) ? "True" : "False") << "\n\n";

    cout << "Deleting element 56 from s4...\n";
    s4.deleteElement(56);
    cout << "set s4 = { ";
    s4.printSet();
    cout << "}.\n\n";

    cout << "Is s4 empty? " << (s4.isEmpty() ? "Yes" : "No") << "\n\n";
    cout << "Is s4 full? " << (s4.isFull() ? "Yes" : "No") << "\n\n";
    cout << "Cardinality of s4 is: " << s4.cardinalityIs() << "\n\n";

    return 0;
}

//================================================

#ifndef INTEGERSET_H
#define INTEGERSET_H

#include <iostream>
using namespace std;

const int SIZE = 101;

class IntegerSet
{
private:
    int a[SIZE];
```

```
public:
   IntegerSet();
   IntegerSet(int *, int);
   IntegerSet unionOfSets(const IntegerSet& s2) const;
   IntegerSet intersectionOfSets(const IntegerSet& s2) const;
   bool isEqualTo(const IntegerSet& s2) const;
   void insertElement(int n);
   void deleteElement(int n);
   bool isEmpty() const;
   bool isFull() const;
   int cardinalityIs() const;
   void printSet() const;
};

#endif

//=================================================

#include "IntegerSet.h"

IntegerSet::IntegerSet()
{
   for (int i = 0; i < SIZE; i++)
      a[i] = 0;
}

IntegerSet::IntegerSet(int *ptr, int arraySize)
{
   for (int i = 0; i < SIZE; i++)
      a[i] = 0;

   for (int i = 0; i < arraySize; i++)
      a[(ptr[i])] = 1;
}

IntegerSet IntegerSet::unionOfSets(const IntegerSet& s2) const
{
   IntegerSet s3;
   for (int i = 0; i < SIZE; i++)
   {
      if (a[i] == 1 || s2.a[i] == 1)
         s3.a[i] = 1;
   }
   return s3;
```

```
}

void IntegerSet::printSet() const
{
   for (int i = 0; i < SIZE; i++)
   {
      if (a[i] == 1)
         cout << i << " ";
   }
}

IntegerSet IntegerSet::intersectionOfSets(const IntegerSet& s2) const
{
   IntegerSet s3;

   for (int i = 0; i < SIZE; i++)
   {
      if (a[i] == 1 && s2.a[i] == 1)
         s3.a[i] = 1;
   }

   return s3;
}

void IntegerSet::insertElement(int n)
{
         // Check validity
         if (n < SIZE && n >= 0)
                  a[n] = 1;
}

bool IntegerSet::isEqualTo(const IntegerSet& s3) const
{
   for (int i = 0; i < SIZE; i++)
   {
      if (a[i] != s3.a[i])
         return false;
   }
   return true;
}

void IntegerSet::deleteElement(int n)
{
   // Check validity
```

```
    if (n < SIZE && n >= 0)
        a[n] = 0;
}

bool IntegerSet::isEmpty() const
{
    for (int i = 0; i < SIZE; i++)
    {
        if (a[i] == 1)
            return false;
    }
    return true;
}

bool IntegerSet::isFull() const
{
    for (int i = 0; i < SIZE; i++)
    {
        if (a[i] == 0)
            return false;
    }
    return true;
}

int IntegerSet::cardinalityIs() const
{
    int count = 0;
    for (int i = 0; i < SIZE; i++)
    {
        if (a[i] == 1)
            count++;
    }
    return count;
}

//==================================================

/*OUTPUT:
Default set (s1):  { }.

Inserting 50, 100, 1, 0, 34, 56 to s1

Now, S1 = { 0 1 34 50 56 100 }.
```

Using array as parameter, set s2 = { 3 34 56 76 }.

Union of s1 and s2, set s3 = { 0 1 3 34 50 56 76 100 }.

Intersection of s1 and s2, set s3 = { 34 56 }.

Inserting 2 elements (34 and 56) into s4...
set s4 = { 34 56 }.

s2 == s3 = False

s3 == s4 = True

Deleting element 56 from s4...
set s4 = { 34 }.

Is s4 empty? No

Is s4 full? No

Cardinality of s4 is: 1

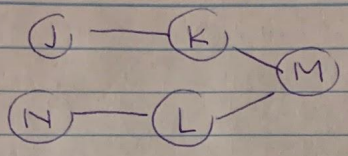
*/
//==

ANSWERS HW_10b #1-5 BELOW

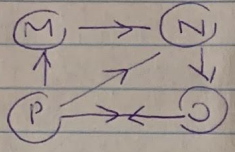
Ethan Nguyen

HW 10

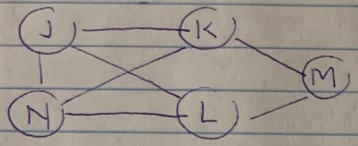
1)



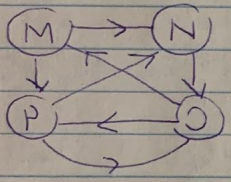
2)



3)



4)



5A) Adjacency matrix

	A	B	C	D	E
A	0	200	600	0	1100
B	200	0	280	300	700
C	600	0	0	1400	0
D	0	0	0	0	900
E	0	0	400	900	0

5B) A → B, 200, → C, 600, → E, 1100
B → A, 200, → C, 280, → E, 700
C → A, 600, → D, 1400
D → E, 800, →
E → C, 400, → D, 900

5C) A to E => A B E (900)