ECE 503 – In-Class Assignment

Create class IntegerSet for which each object can hold integers in the range 0 through 100. A set is represented internally as an array of ones and zeros. Array element a[ i ] is 1 if integer i is in the set. Array element a[ j ] is 0 if integer j is not in the set. The default constructor initializes a set to the so-called “empty-set,” i.e., a set whose array representation contains all zeros.

Class IntegerSet has the following member functions:

1. inputSet: read values from user
2. unionOfSets: create a third set that is the set union of two existing sets (i.e., an element of the third array’s is set to 1 if that element is 1 in either or both of the existing sets, and an element of the third set’s array is set to 0 if that element is 0 in each of the existing sets).
3. intersectionOfSets: create a third set that is the set intersection of two existing sets (i.e., an element of the third array’s is set to 1 if that element is 1 in both of the existing sets, and an element of the third set’s array is set to 0 if that element is 0 in either of the existing sets).
4. printSet: print a set as a list of numbers separated by spaces in between a pair of curly braces. Print only those elements which are present in the set (i.e., their position in the array has a value of 1).

The following is a template that you need to use for your program - you need to fill in the necessary components in the template to make it work:

// IntegerSet.h

// Header file for class IntegerSet

#ifndef INTEGER\_SET\_H

#define INTEGER\_SET\_H

class IntegerSet

{

public:

IntegerSet( ); // constructor

/\* Write a member funcion prototype for UnionOfSets \*/

void inputSet(); // read values from user

void printSet() const;

private:

int set[ 101 ]; // range of 0 - 100

// determines a valid entry to the set

int validEntry( int x) const

{

return ( x >= 0 && x <= 100 );

} // end function validEntry

};

----------------------------------------------------------------------------------------------------

// IntegerSet.cpp

// Member-function definitions for class IntegerSet.

#include <iostream>

#include <iomanip>

using namespace std;

/\* Write include directive for IntegerSet.h here \*/

// constructor

IntegerSet::IntegerSet()

{

for ( int i = 0; i < 101; i++ ) set[ i ] = 0;

} // end IntegerSet constructor

// input a set from the user

void IntegerSet::inputSet()

{

int number;

do

{

cout << "Enter an element (-1 to end): ";

cin >> number;

if ( validEntry( number ) )

set[ number ] = 1;

else if ( number != -1 )

cout << "Invalid Element\n";

} while ( number != -1 ); // end do...while

cout << "Entry complete\n";

} // end function inputSet

// prints the set to the output stream

void IntegerSet::printSet() const

{

cout << “{ “;

for (int u = 0; u < 101; u++ )

if ( set[ u ] ) cout << u << “ “;

cout << "}" << endl;

} // end function printSet

/\* Write definition for unionOfSets \*/

/\* Write definition for intersectionOfSets \*/

----------------------------------------------------------------------------------------------------

// SetTest.cpp

// Driver program for class IntegerSet.

#include <iostream>

using namespace std;

#include "IntegerSet.h" // IntegerSet class definition

int main()

{

IntegerSet a;

IntegerSet b;

IntegerSet c;

IntegerSet d;

cout << "Enter set A:\n";

a.inputSet();

cout << “\nSet A is:\n”;

a.printSet();

cout << "\nEnter set B:\n";

b.inputSet();

cout << “\nSet B is:\n”;

b.printSet();

/\* Write call to unionOfSets for object a, passing b as argument and assigning the result to c \*/

cout << "\nUnion of A and B is:\n";

c.printSet();

/\* Write call to intersectionOfSets for object a, passing b as argument and assigning the result to d \*/

cout << "\nIntersection of A and B is:\n";

d.printSet();

return 0;

}

----------------------------------------------------------------------------------------------------

The following is a sample output of your program:

Enter set A:

Enter an element (-1 to end): 45

Enter an element (-1 to end): 76

Enter an element (-1 to end): 34

Enter an element (-1 to end): 6

Enter an element (-1 to end): -1

Entry complete

Set A is:

{ 6 34 45 76 }

Enter set B:

Enter an element (-1 to end): 34

Enter an element (-1 to end): 8

Enter an element (-1 to end): 93

Enter an element (-1 to end): 45

Enter an element (-1 to end): -1

Entry complete

Set B is:

{ 8 34 45 93 }

Union of A and B is:

{ 6 8 34 45 76 93 }

Intersection of A and B is:

{ 34, 45 }