UMass

Electrical and Computer
Engineering Department

ECE 264, **Object-Oriented Software Development**

Lab Assignment 9

Lab 8 :: 100 points (see Grading Notes for details) ::
Wednesday lab session (April 10) Due April 12, Friday by 5:00 pm
Monday lab session (April 15) Rescheduled to April 17, Due April 19, Friday, by 5:00 pm

1. Lab Objectives

This lab was designed to reinforce programming concepts from Chapter 10 of *C++ How To Program, 8th Edition*.
In this lab, you will practice:

- Using classes to create a data type, IntegerSet, capable of storing a set of integers.
- Using dynamic memory allocation with the new and delete operators.

2. Deliverables

Create “lab9” sub-directory on your M:\ drive. Submit your file to this sub-directory on the M:\ drive. Call your project *lab9_IntegerSet* respectively. You should place all the source files (.h and .cpp) on the”lab9” sub-directory. Failure to meet this specification will reduce your grade, as described in the ECE 264 lab grading handout, which you are strongly encouraged to read before starting the lab.

3. Description of the Problem

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. Provide member functions for the common set operations. For example, a unionOfSets member function (already provided) creates a third set that is the set-theoretic 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).

Provide an intersectionOfSets member function which creates a third set which is the set-theoretic intersection of two existing sets (i.e., an element of the third set’s array is set to 0 if that element is 0 in either or both of the existing sets, and an element of the third set’s array is set to 1 if that element is 1 in each of the existing sets). An insertElement member function (already provided) inserts a new integer *k* into a set (by setting a[k] to 1). Provide a deleteElement member function that deletes integer *m* (by setting a[m] to 0).
A printSet member function (already provided) prints a set as a list of numbers separated by spaces. Print only those elements which are present in the set (i.e., their position in the array has a value of 1). Print --- for an empty set.

Provide an isEqualTo member function that determines whether two sets are equal.

Provide an additional constructor that receives an array of integers and the size of that array and uses the array to initialize a set object.

Now write a driver program to test your IntegerSet class. Instantiate several IntegerSet objects. Test that all your member functions work properly.

```
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

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

Union of A and B is:
{   6   8  34  45  76  93   }
Intersection of A and B is:
{  34  45   }
Set A is not equal to set B

Inserting 77 into set A...
Set A is now:
{   6  34  45  76  77   }

Deleting 77 from set A...
Set A is now:
{   6  34  45  76   }
Invalid insert attempted!
Invalid insert attempted!

Set e is:
{   1   2   9  25  45  67  99 100   }
```

Template

```
// Lab 2: IntegerSet.h
// Header file for class IntegerSet
#ifndef INTEGER_SET_H
#define INTEGER_SET_H

class IntegerSet
{
public:
   // default constructor
   IntegerSet()
   {
```

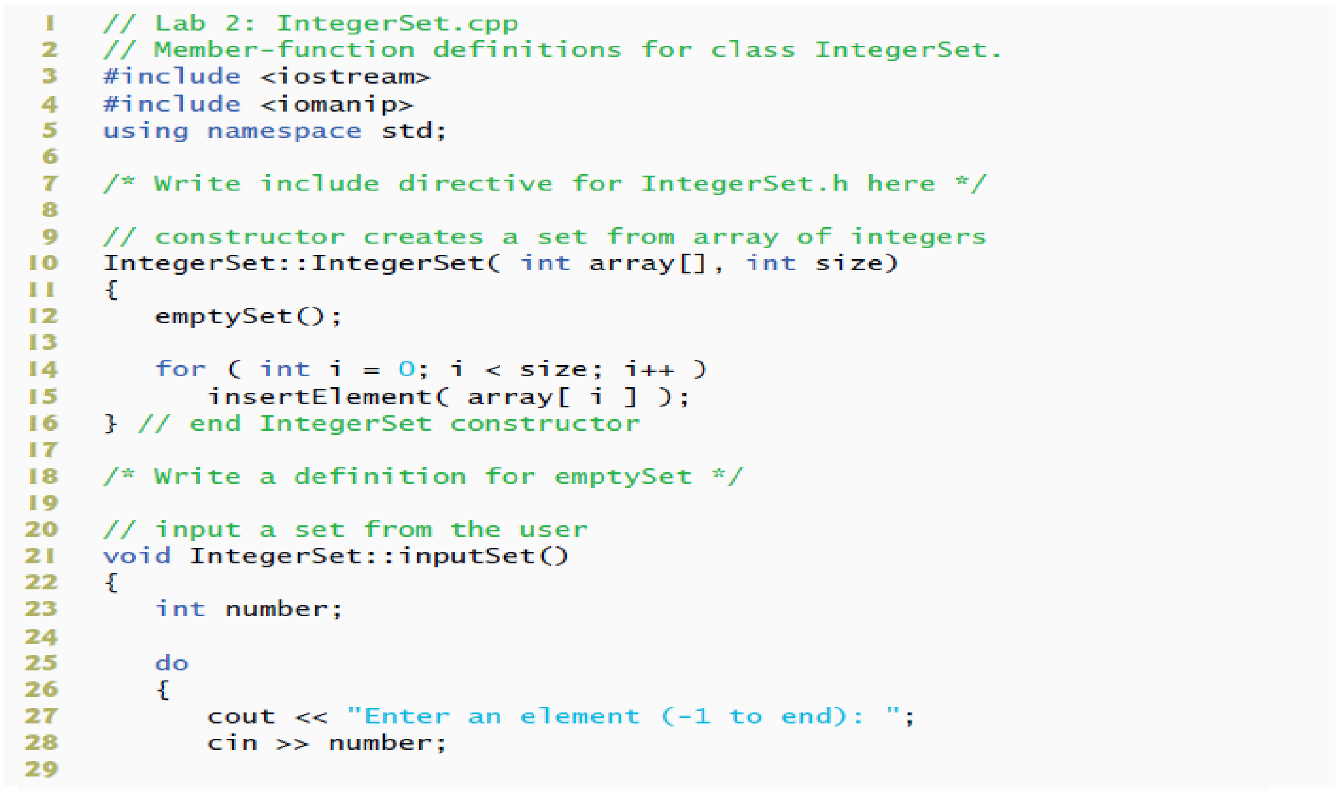
Fig. L 10.4 | Contents of integerset.h. (Part 1 of 2.)

Lab Assignment 9

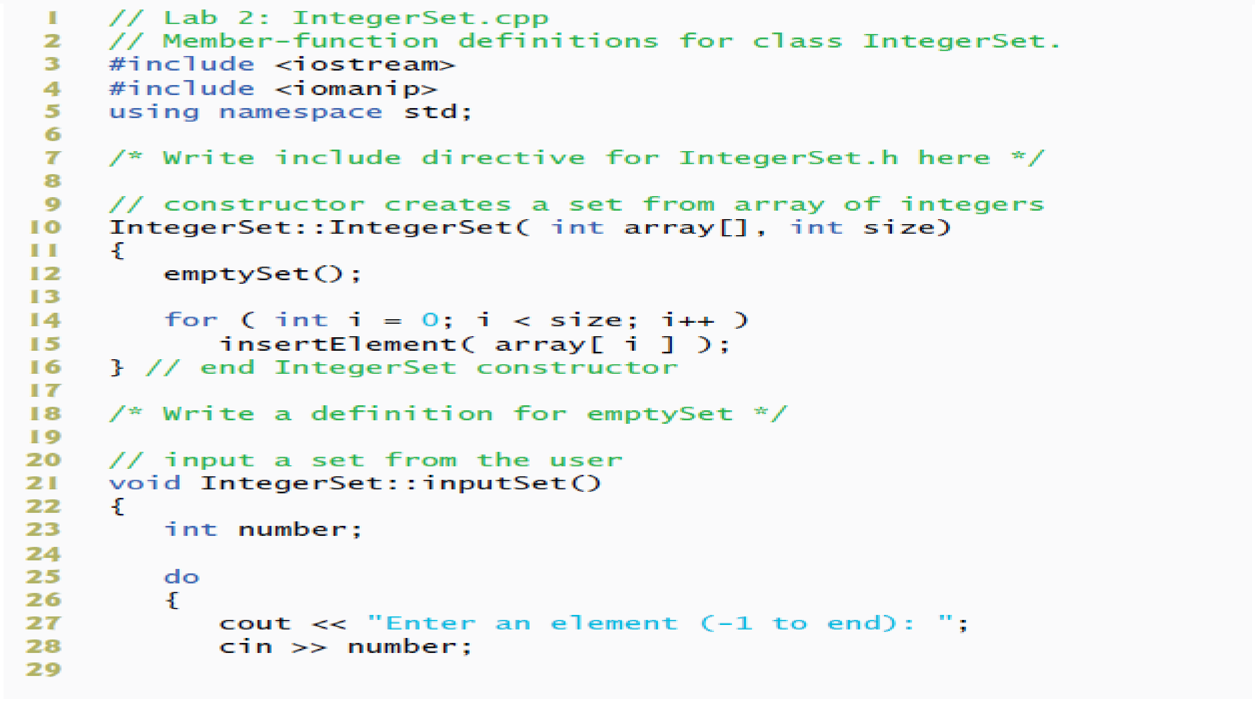
```
      /* Write call to emptySet */
   } // end IntegerSet constructor

   IntegerSet( int [], int ); // constructor that takes an initial set
   IntegerSet unionOfSets( const IntegerSet& );
   /* Write a member funcion prototype for intersectionOfSets */
   void emptySet(); // set all elements of set to 0
   void inputSet(); // read values from user
   void insertElement( int );
   /* Write a member function prototype for deleteElement */
   void printSet() const
   /* Write a member function prototype for isEqualTo */
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
}; // end class IntegerSet

#endif
```

Fig. L 10.4 | Contents of `integerset.h`. (Part 2 of 2.)

Lab Assignment 9



```
// Lab 2: IntegerSet.cpp
// Member-function definitions for class IntegerSet.
#include <iostream>
#include <iomanip>
using namespace std;

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

// constructor creates a set from array of integers
IntegerSet::IntegerSet( int array[], int size)
{
   emptySet();

   for ( int i = 0; i < size; i++ )
      insertElement( array[ i ] );
} // end IntegerSet constructor

/* Write a definition for emptySet */

// input a set from the user
void IntegerSet::inputSet()
{
   int number;

   do
   {
      cout << "Enter an element (-1 to end): ";
      cin >> number;

```



```
// Lab 2: IntegerSet.cpp
// Member-function definitions for class IntegerSet.
#include <iostream>
#include <iomanip>
using namespace std;

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

// constructor creates a set from array of integers
IntegerSet::IntegerSet( int array[], int size)
{
   emptySet();

   for ( int i = 0; i < size; i++ )
      insertElement( array[ i ] );
} // end IntegerSet constructor

/* Write a definition for emptySet */

// input a set from the user
void IntegerSet::inputSet()
{
   int number;

   do
   {
      cout << "Enter an element (-1 to end): ";
      cin >> number;

```

Fig. L 10.5 | Contents of integerset.cpp. (Part 1 of 3.)

Lab Assignment 9

```
      if ( validEntry( number ) )
         set[ number ] = 1;
      else if ( number != -1 )
         cerr << "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
{
   int x = 1;
   bool empty = true; // assume set is empty

   cout << '{';

   for (int u = 0; u < 101; u++ )
   {
      if ( set[ u ] )
      {
         cout << setw( 4 ) << u << ( x % 10 == 0 ? "\n" : "" );
         empty = false; // set is not empty
         ++x;
      } // end if
   } // end for

   if ( empty )
      cout << setw( 4 ) << "---"; // display an empty set

   cout << setw( 4 ) << "}" << '\n';
} // end function printSet

// returns the union of two sets
IntegerSet IntegerSet::unionOfSets( const IntegerSet &r )
{
   IntegerSet temp;

   // if element is in either set, add to temporary set
   for ( int n = 0; n < 101; n++ )
      if ( set[ n ] == 1 || r.set[ n ] == 1 )
         temp.set[ n ] = 1;

   return temp;
} // end function unionOfSets

/* Write definition for intersectionOfSets */

// insert a new integer into this set
void IntegerSet::insertElement( int k )
{
   if ( validEntry( k ) )
      set[ k ] = 1;
   else
      cerr << "Invalid insert attempted!\n";
} // end function insertElement
```

Fig. L 10.5 | Contents of intergerSet.cpp (Part 2 of 3)

Lab Assignment 9

```

/* Write definition for deleteElement */

/* Write definition for isEqualTo */

// determines if two sets are equal
bool IntegerSet::isEqualTo( const IntegerSet &r ) const
{
   for ( int v = 0; v < 101; v++ )
      if ( set[ v ] != r.set[ v ] )
         return false; // sets are not-equal

   return true; // sets are equal
} // end function isEqualTo
```

Fig. L 10.5 | Contents of `integerset.cpp`. (Part 3 of 3.)

```
// Lab 2: 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 << "\nEnter set B:\n";
   b.inputSet();
   /* Write call to unionOfSets for object a, passing
      b as argument and assigning the result to c */
   /* Write call to intersectionOfSets for object a,
      passing b as argument and assigning the result to d */
   cout << "\nUnion of A and B is:\n";
   c.printSet();
   cout << "Intersection of A and B is:\n";
   d.printSet();

   if ( a.isEqualTo( b ) )
      cout << "Set A is equal to set B\n";
   else
      cout << "Set A is not equal to set B\n";

   cout << "\nInserting 77 into set A...\n";
   a.insertElement( 77 );
   cout << "Set A is now:\n";
   a.printSet();

```

Fig. L 10.6 | Contents of `SetTest.cpp`. (Part 1 of 2.)

Lab Assignment 9

```
    cout << "\nDeleting 77 from set A...\n";
    a.deleteElement( 77 );
    cout << "Set A is now:\n";
    a.printSet();

    const int arraySize = 10;
    int intArray[ arraySize ] = { 25, 67, 2, 9, 99, 105, 45, -5, 100, 1 };
    IntegerSet e( intArray, arraySize );

    cout << "\nSet e is:\n";
    e.printSet();

    cout << endl;
} // end main
```

Fig. L 10.6 | Contents of SetTest.cpp. (Part 2 of 2.)

Problem-Solving Tips

1. Member function intersectionOfSets must return an IntegerSet object. The object that invokes this function and the argument passed to the member function should not be modified by the operation. intersectionOfSets should iterate over all integers an IntegerSet could contain (1–100) and add those integers that both IntegerSets contain to a temporary IntegerSet that will be returned.

2. Member function deleteElement should first verify that its argument is valid by calling utility function validEntry. If so, the corresponding element in the set array should be set to 0; otherwise, display an error message.

3. Member function isEqualTo should iterate over all integers an IntegerSet could contain and (1–100). If any integer is found that is in one set but not the other, return false; otherwise return true.

4. Testing Your Program

- For this program, there is no user input so the only way to test your program is to run it and see if it displays all of the information correctly.
- In all of your programs, but especially a program where there isn't any user input, you should focus on making the output easy to read. One of the most difficult things for a user of your program to deal with is poorly formatted output. The easier your output is to read, the easier it is to identify the relevant information that you're producing.