COMSC 200

Summer 2024

Programming Assignment 7

Worth 12.5 points (1.25% of your grade)

DUE: Thursday, 7/4/24 by 11:59 P.M. on Canvas

Late Pass Deadline: Sunday, 7/7/24 by 11:59 P.M. on Canvas

Start by downloading the **200_assign7.cpp** file from the Programming Assignment 7 folder on Canvas*

*This file contains the **complete** application program – it should **NOT** be modified in any way. The IntegerSet.h and IntegerSet.cpp class files that you implement will need to work with the 200_assign7.cpp application program file **as is**, without **any** changes being made to it.

*Also, please do **NOT** need to submit the 200_assign7.cpp file as part of your submission. Your IntegerSet.h and IntegerSet.cpp files will be run with the 200_assign7.cpp file that I have provided for you. **So it's absolutely critical that your class code works with the application program that I have provided for you.**

Please continue to use the same **naming convention** as before, where each filename should contain both your first name and your last name. If your first name is "James" and your last name is "Smith", then your header file should be named James_Smith_IntegerSet.h, your cpp file should be named James_Smith_IntegerSet.cpp, and your pdf file should be named James_Smith.pdf.

Your submission should consist of **three (3)** files:

first-name_last-name_IntegerSet.h (class specification file) – This file should declare all of the member variables and the **prototypes** for constructors and functions. None of the constructors/functions should be implemented in this file.

first-name_last-name_IntegerSet.cpp (class implementation file) – This file should contain the **implementation** for all of the constructors and functions.

And the first-name_last-name.pdf file should contain your five (5) sample runs.

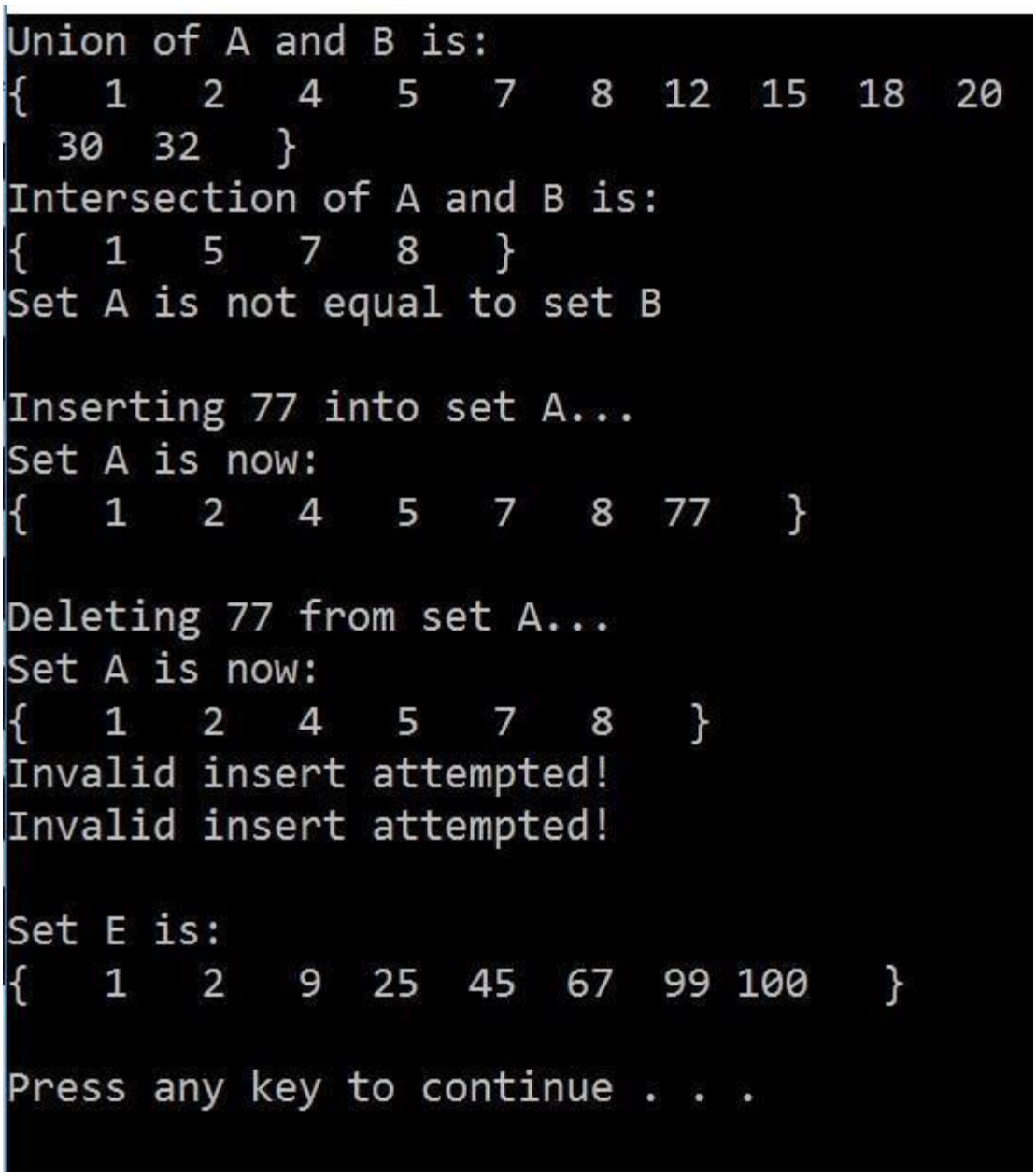
Comments – worth 1.25 points (10%) of your programming assignment grade:

Your program should have at least **ten (10)** different detailed comments explaining the different parts of your program. Each individual comment should be, at a minimum, a sentence explaining a particular part of your code. You should make each comment as detailed as necessary to fully explain your code. You should also number each of your comments (i.e., comment 1, comment 2, etc.).

Sample Runs – worth 1.25 points (10%) of your programming assignment grade:

You should submit screenshots of at least **five (5)** different sample runs of your program. You should also number each of your sample runs (i.e., sample run 1, sample run 2, etc.). **NOTE: Your sample runs should be different from my sample runs shown in this write-up for the programming assignment.** Each sample run should follow this format:

```
Enter set A:
Enter an element (-1 to end): 1
Enter an element (-1 to end): 5
Enter an element (-1 to end): 4
Enter an element (-1 to end): 7
Enter an element (-1 to end): 8
Enter an element (-1 to end): 2
Enter an element (-1 to end): -1
Entry complete

Enter set B:
Enter an element (-1 to end): 20
Enter an element (-1 to end): 1
Enter an element (-1 to end): 5
Enter an element (-1 to end): 8
Enter an element (-1 to end): 12
Enter an element (-1 to end): 15
Enter an element (-1 to end): 18
Enter an element (-1 to end): 7
Enter an element (-1 to end): 30
Enter an element (-1 to end): 32
Enter an element (-1 to end): -1
Entry complete
```



For this programming assignment you will be creating a class named IntegerSet for which each object can hold integers in the range **0 through 100**. Represent the set internally as a vector of bool values. Element a[i] is true if integer ***i*** is in the set. Element a[j] is false if integer ***j*** is not in the set. The default constructor initializes a set to the so-called “empty set,” i.e., a set for which all elements contain false.

a. Provide member functions for the common set operations. For example, provide a unionOfSets member function that creates a third set that is the set-theoretic union of two existing sets (i.e., an element of the result is set to true if that element is true in either or both of the existing sets, and an element of the result is set to false if that element is false in each of the existing sets).

b. 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 result is set to false if that element is false in either or both of the existing sets, and an element of the result is set to true if that element is true in each of the existing sets).

c. Provide an insertElement member function that places a new integer *k* in to a set by setting a[k] to true. Provide a deleteElement member function that deletes integer *m* by setting a[m] to false.

d. Provide a printSet member function that prints a set as a list of numbers separated by spaces. Print only those elements that are present in the set (i.e., their position in the vector has a value of true). Print -- for an empty set.

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

f. 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.

g. Provide the inputSet member function and the second constructor that are used in the application program as well.

   The valid range of number is 0 – 100; any number outside of this range is considered invalid. The number -1 is treated as special case: it is used by the user to signal that they are done entering elements for the set.

   Also, remember one of the member variables in the IntegerSet class is a **vector of bools**. Suppose this vector is named set with the following sample values:

   set[0] = true (this means the number 0 is in the set) set[1]
   = false (this means the number 1 is **NOT** in the set) set[2]
   = false (this means the number 2 is **NOT** in the set)
   …
   set[100] = true (this means the number 100 is in the set)

   As another example, suppose the set consists of the values {0, 50, 100}. Then:

set[0] = true
set[50] = true
set[100] = true

And all other indexes in set would store false.

Finally, it is also possible for a set to be empty. An **empty set** is a set in which all indexes are set to false:

set[0] = false
set[1] = false
…
set[100] = false

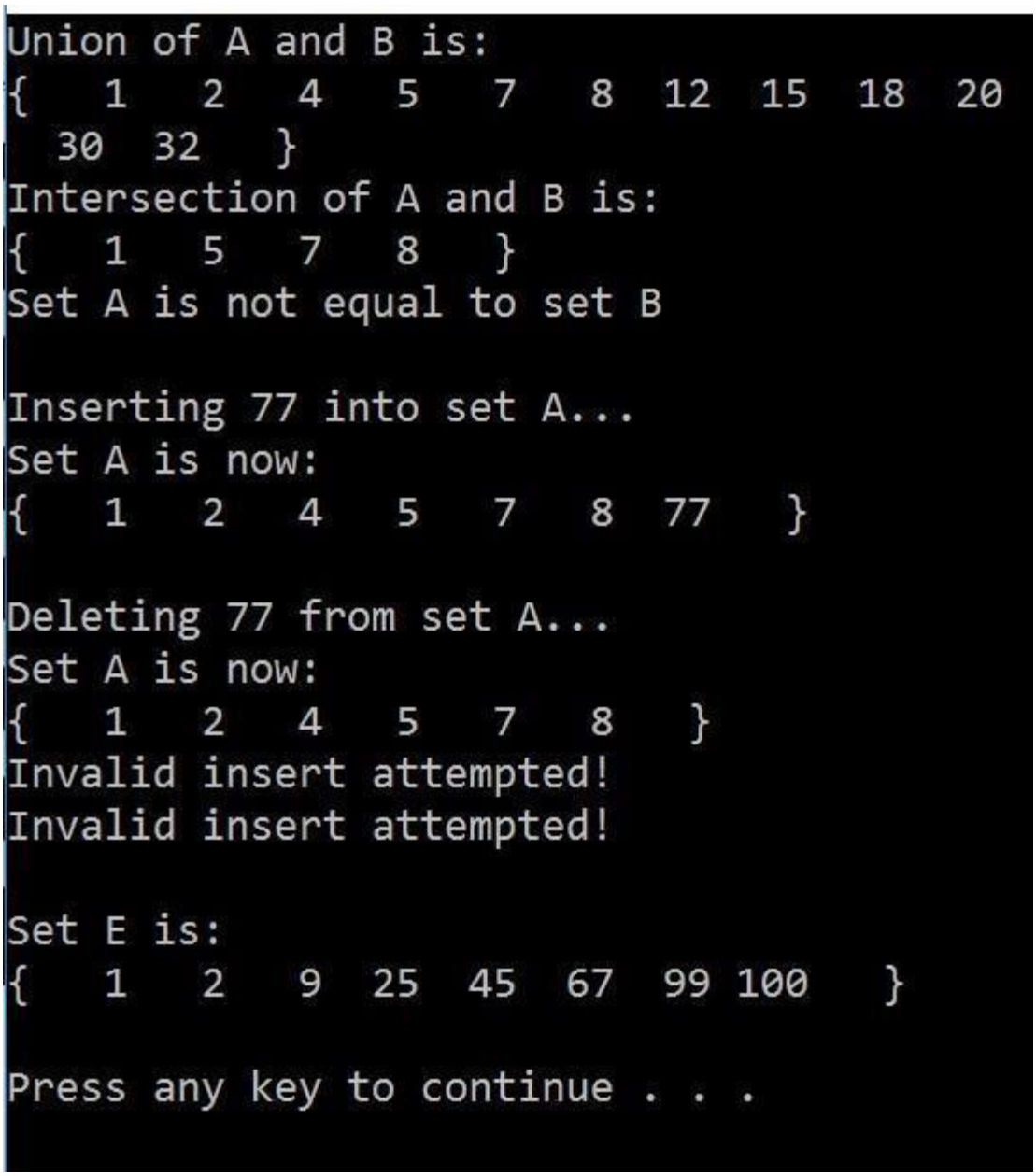
An empty set is a set in which **NONE** of the numbers 0 – 100 occur in the set. In mathematics an empty set is represented as {}. In the program {--} will be printed out to represent an empty set.

The size of the vector is 101, with indexes 0 – 100. For each index, the Boolean valued stored (true or false) indicates whether that number is in the set or not.

Make sure to **thoroughly** test your program. You should get the **same exact output** as shown in the screenshots below for all six sample runs.

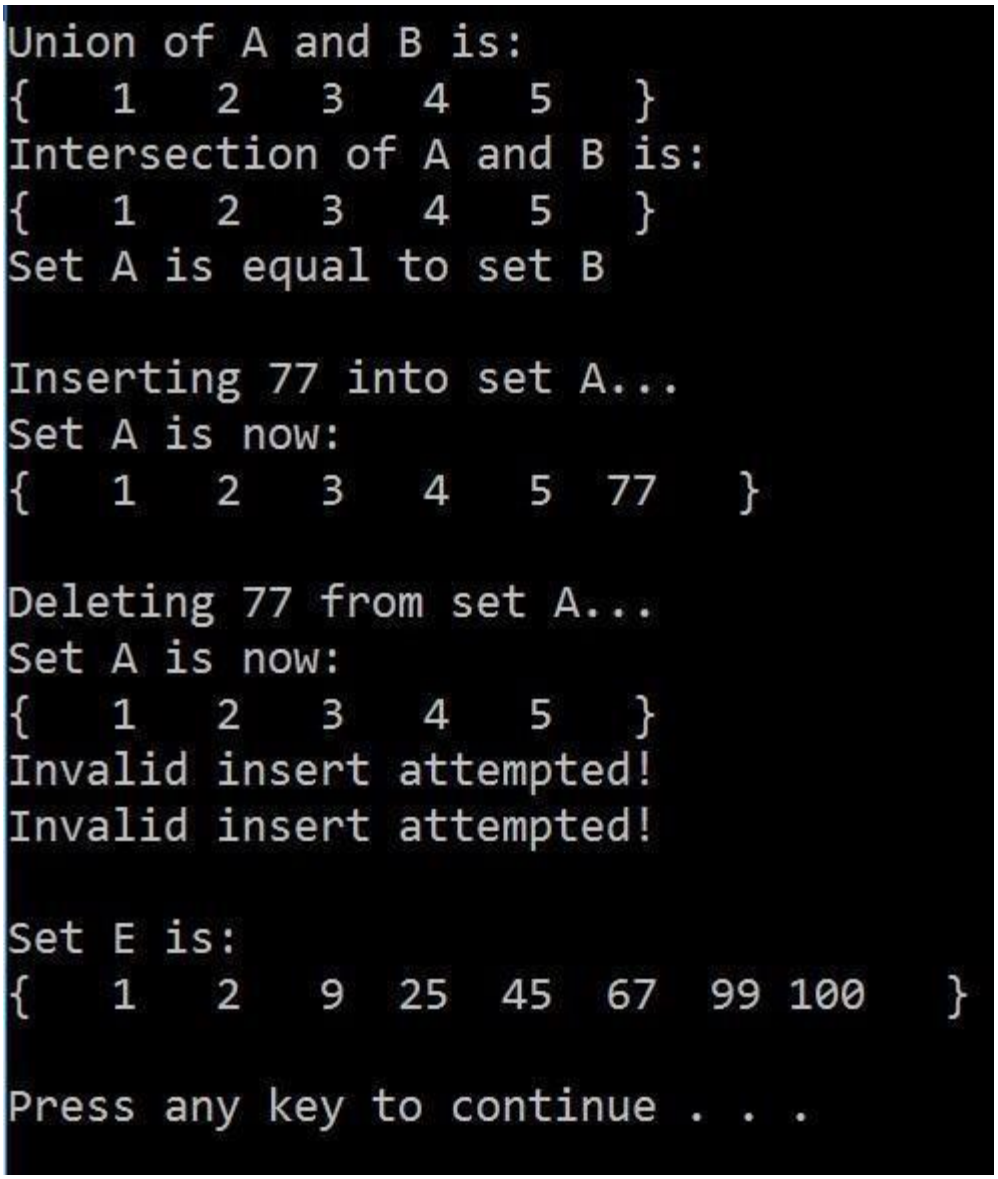
Sample run 1 (using the 200_assign7.cpp application program file):

```
Enter set A:
Enter an element (-1 to end): 1
Enter an element (-1 to end): 5
Enter an element (-1 to end): 4
Enter an element (-1 to end): 7
Enter an element (-1 to end): 8
Enter an element (-1 to end): 2
Enter an element (-1 to end): -1
Entry complete

Enter set B:
Enter an element (-1 to end): 20
Enter an element (-1 to end): 1
Enter an element (-1 to end): 5
Enter an element (-1 to end): 8
Enter an element (-1 to end): 12
Enter an element (-1 to end): 15
Enter an element (-1 to end): 18
Enter an element (-1 to end): 7
Enter an element (-1 to end): 30
Enter an element (-1 to end): 32
Enter an element (-1 to end): -1
Entry complete
```



Sample run 2 (using the 200_assign7.cpp application program file):

```
Enter set A:
Enter an element (-1 to end): 1
Enter an element (-1 to end): 1
Enter an element (-1 to end): 2
Enter an element (-1 to end): 2
Enter an element (-1 to end): 2
Enter an element (-1 to end): 3
Enter an element (-1 to end): 3
Enter an element (-1 to end): 4
Enter an element (-1 to end): 4
Enter an element (-1 to end): 4
Enter an element (-1 to end): 4
Enter an element (-1 to end): 5
Enter an element (-1 to end): 5
Enter an element (-1 to end): -1
Entry complete

Enter set B:
Enter an element (-1 to end): 1
Enter an element (-1 to end): 1
Enter an element (-1 to end): 2
Enter an element (-1 to end): 2
Enter an element (-1 to end): 3
Enter an element (-1 to end): 3
Enter an element (-1 to end): 4
Enter an element (-1 to end): 4
Enter an element (-1 to end): 5
Enter an element (-1 to end): 5
Enter an element (-1 to end): -1
Entry complete
```

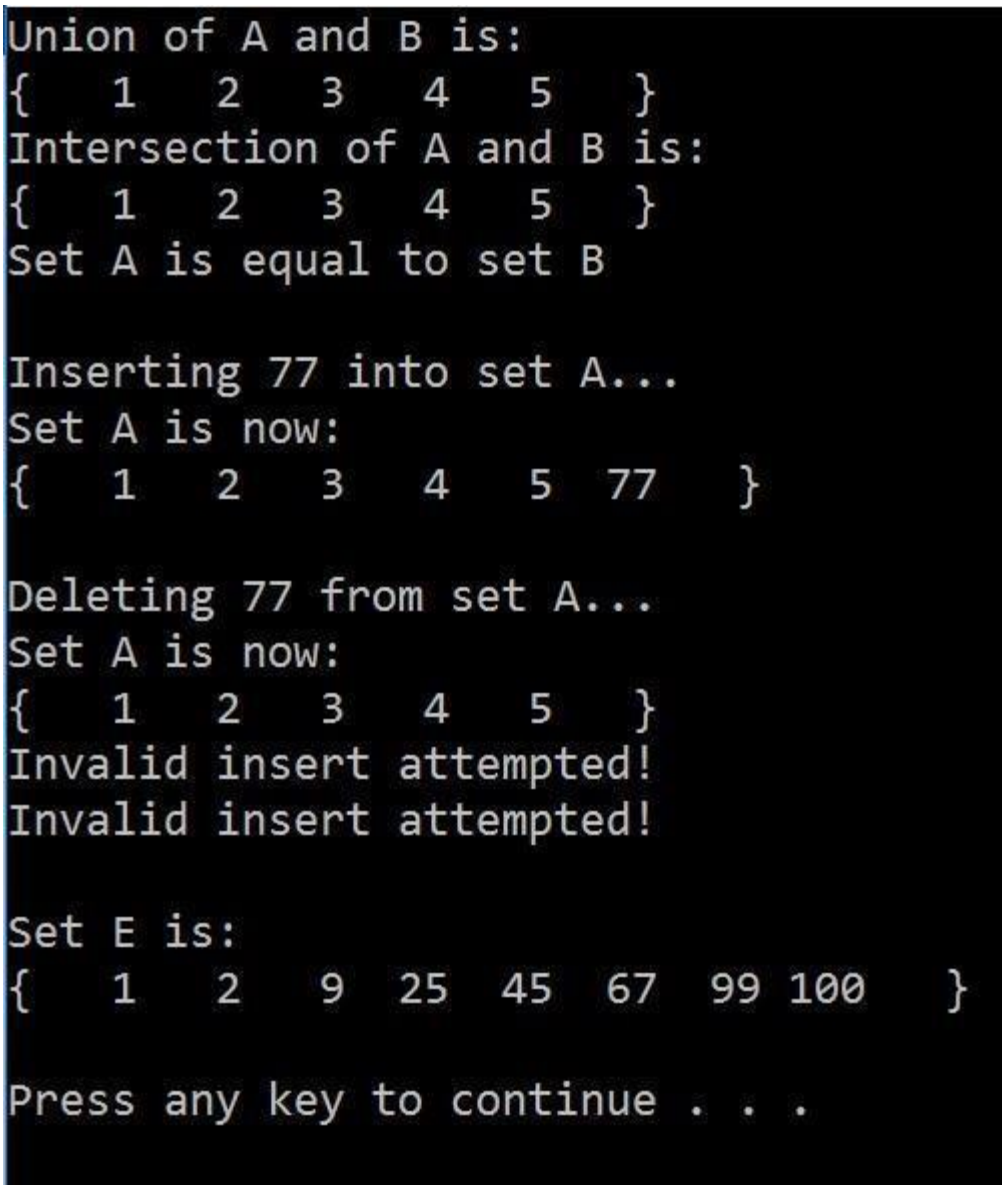


Sample run 3 (using the 200_assign7.cpp application program file):

```
Enter set A:
Enter an element (-1 to end): 10
Enter an element (-1 to end): 20
Enter an element (-1 to end): 30
Enter an element (-1 to end): 40
Enter an element (-1 to end): 50
Enter an element (-1 to end): -1
Entry complete

Enter set B:
Enter an element (-1 to end): 10
Enter an element (-1 to end): 20
Enter an element (-1 to end): 30
Enter an element (-1 to end): 40
Enter an element (-1 to end): 50
Enter an element (-1 to end): 60
Enter an element (-1 to end): -1
Entry complete

Union of A and B is:
{  10  20  30  40  50  60   }
Intersection of A and B is:
{  10  20  30  40  50   }
Set A is not equal to set B

Inserting 77 into set A...
Set A is now:
{  10  20  30  40  50  77   }

Deleting 77 from set A...
Set A is now:
{  10  20  30  40  50   }
Invalid insert attempted!
Invalid insert attempted!

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

Press any key to continue . . .
```

Sample run 4 (using the 200_assign7.cpp application program file):

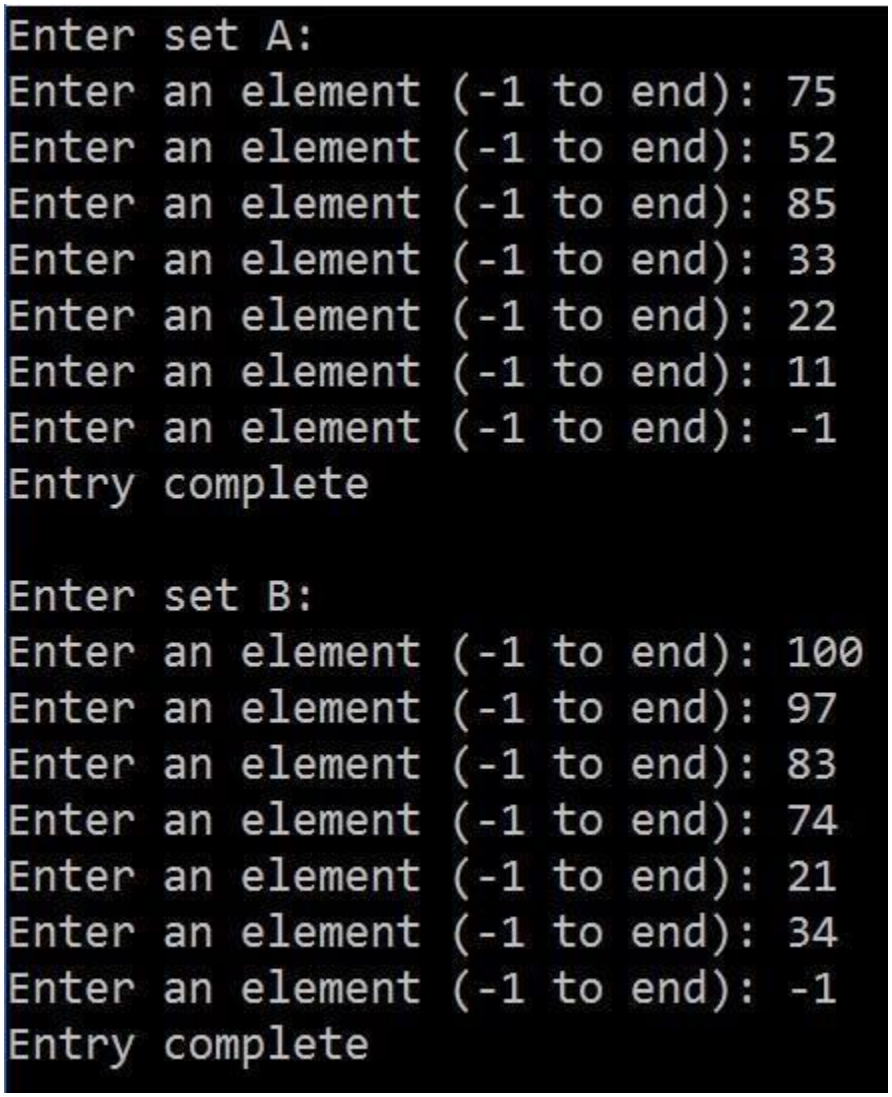
```
Enter set A:
Enter an element (-1 to end): 101
Invalid Element
Enter an element (-1 to end): 1
Enter an element (-1 to end): 2
Enter an element (-1 to end): -5
Invalid Element
Enter an element (-1 to end): 3
Enter an element (-1 to end): 4
Enter an element (-1 to end): 150
Invalid Element
Enter an element (-1 to end): 5
Enter an element (-1 to end): -1
Entry complete

Enter set B:
Enter an element (-1 to end): 1
Enter an element (-1 to end): 1
Enter an element (-1 to end): 1
Enter an element (-1 to end): 900
Invalid Element
Enter an element (-1 to end): 2
Enter an element (-1 to end): 2
Enter an element (-1 to end): 127
Invalid Element
Enter an element (-1 to end): 3
Enter an element (-1 to end): 3
Enter an element (-1 to end): -9
Invalid Element
Enter an element (-1 to end): 4
Enter an element (-1 to end): 4
Enter an element (-1 to end): 5
Enter an element (-1 to end): 5
Enter an element (-1 to end): -1
Entry complete
```



Sample run 5 (using the 200_assign7.cpp application program file):

```
Enter set A:
Enter an element (-1 to end): -1
Entry complete

Enter set B:
Enter an element (-1 to end): 101
Invalid Element
Enter an element (-1 to end): 150
Invalid Element
Enter an element (-1 to end): -6
Invalid Element
Enter an element (-1 to end): -1
Entry complete

Union of A and B is:
{ ---   }
Intersection of A and B is:
{ ---   }
Set A is equal to set B

Inserting 77 into set A...
Set A is now:
{  77   }

Deleting 77 from set A...
Set A is now:
{ ---   }
Invalid insert attempted!
Invalid insert attempted!

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

Press any key to continue . . .
```

Sample run 6 (using the 200_assign7.cpp application program file):



```
Union of A and B is:
{  11  21  22  33  34  52  74  75  83  85
  97 100   }
Intersection of A and B is:
{ ---   }
Set A is not equal to set B

Inserting 77 into set A...
Set A is now:
{  11  22  33  52  75  77  85   }

Deleting 77 from set A...
Set A is now:
{  11  22  33  52  75  85   }
Invalid insert attempted!
Invalid insert attempted!

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

Press any key to continue . . .
```