PIC 10B Lectures 1 and 2 Spring 2015
Homework Assignment #2

Due Wednesday, April 15, 2015 by 6:00pm.

Objectives:

1. To overload C++ operators.

Introduction:

A set is a collection of values of the same type. For example, {1 2 3} is a set whose elements are 1, 2, and 3. A set is not allowed to contain duplicate elements. For example, {1 1} is the set {1} of size 1. You are going to define a Set class which holds a collection of integer values as well as some operations on sets. We will define these operations by overloading C++ operators. The specifications for the assignment are below:

The class `Set` will have the following characteristics:

Attributes:
- elements (a vector of `int` values)

Constructors:
- `Set()             // creates an empty Set`
- `Set(int element) // creates a Set having only the given`
  `                  // element`

Behaviors (public members):
- A predicate function `contains(int element)` which returns `true` if the calling Set object contains the given element, `false` otherwise. It should not change the calling object.
- A function `size()` which returns how many elements the set contains as an unsigned int. It should not change the calling object.

Operators (as public members):
- `operator[]` takes in an index and returns the element at the given index. It should not change the calling object. If the given index is out of bounds, print an error message and call `exit(1)` to terminate the program.
- `operator+=` computes the union of the calling object Set and the Set object passed in and stores the resulting union as the calling object. The union of two sets is the set formed by including the elements of both sets into one big set. For example, the set union of {1 2} and {4 3 2} is the set {1 2 4 3}.
- `operator-=` computes the set difference of the calling object Set from the Set passed in and stores the resulting set difference as the calling object.

The difference of set A from set B is the set formed by the elements of A which are not in set B. For example, {1 2 3 4 5} – { 2 5 6 } is {1 3 4}.
- `operator*=` computes the set intersection of the calling Set object and the given Set object passed in and stores the resulting intersection as the calling object. The intersection of two sets is the set formed by the elements that both sets have in common. For example, the intersection of {1 2 3 4} and {2 4 5} is {2 4}.

Operators (as friends):
- operator<< should output the Set object as a list of elements separated only by spaces. Use { and } to delimit the beginning and end of the Set. For example, {5 -9 3 2} would be the output of a Set. Be sure to return the stream at the end.
- operator>> should read a Set that is presented in the same format as mentioned above (see operator <<). Be sure to return the stream at the end.

Binary Operators (as nonmember nonfriends):
- `operator+` returns the union of the two given Set objects. This function should pass the operand Set objects in by reference and return the union Set by value. Be sure to reuse what code you already have.
- `operator-` returns the difference of the second Set from the first Set operand. This function should pass the operand Set objects in by reference and return the Set difference by value.
  Example: {1 2 3 4 5} – { 2 5 6 } is {1 3 4}. It is not {2 6}.
- `operator*` returns the intersection of the two given Set operands. This function should pass the operand Set objects in by reference and return the Set intersection by value.
- `operator<=` returns true if every element of the given first Set is contained in the second Set, false otherwise.
- `operator>=` returns true if every element of the given second Set is contained in the first Set, false otherwise.
- `operator==` returns true if each of the given two Sets is contained in the other one, false otherwise
- `operator!=` returns the opposite value of operator==
- `operator<` returns true if every element of the given first Set is contained in the second Set and the two sets are not equal, false otherwise.
- `operator>` returns true if every element of the given second Set is contained in the first Set and the two sets are not equal, false otherwise.

Be sure to not “reinvent any wheels” when defining these operators. Try to use the Set operators you already defined if possible.

Directions:

1. Create the project called “Hw2” in a solution called “Homework” using Microsoft Visual Studio 2012. All header (.h) and source (.cpp) files inside the project should contain the following header:

```
/*
   <Your Name>                    PIC 10B Intermediate Programming
   ID: <Your Student ID>           Spring 2015
   Email: <Your Email Address>     Assignment #2
   Section: <Your Section # eg 1A>
   Honesty Pledge:

      I, <Your Name Here>, pledge that this is my own independent work,
   which conforms to the guidelines of academic honesty as described in
   the course syllabus.

   List of known bugs:  <Known bugs, if any>
*/
```

2. Download the following file and add it to your project: SetApp.cpp. You are not allowed to modify any code in this file.

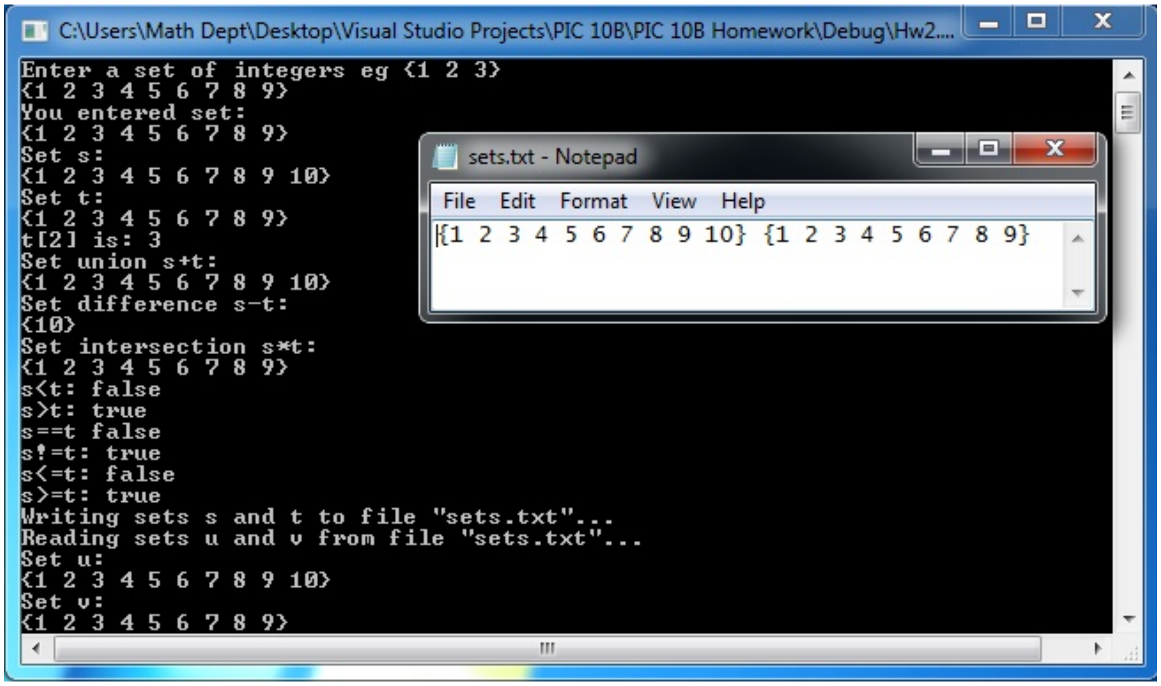
3. Define and implement the Set class as described above. Create a separate interface file Set.h and implementation file Set.cpp for your class. Declare your nonmember operators in Set.h outside the class definition. Define all operators in Set.cpp.

4. When you have completed your project, be sure to
   - make sure your program compiles in Visual Studio 2012.
   - run your program to make sure it works correctly
   - upload your source code files
     - Set.h
     - Set.cpp

     using the CCLE website. No hardcopies will be collected.
   - visually verify that your source code was submitted correctly by clicking on the links to those files on the CCLE page after submission.

Some screenshots (all the same session) are below:

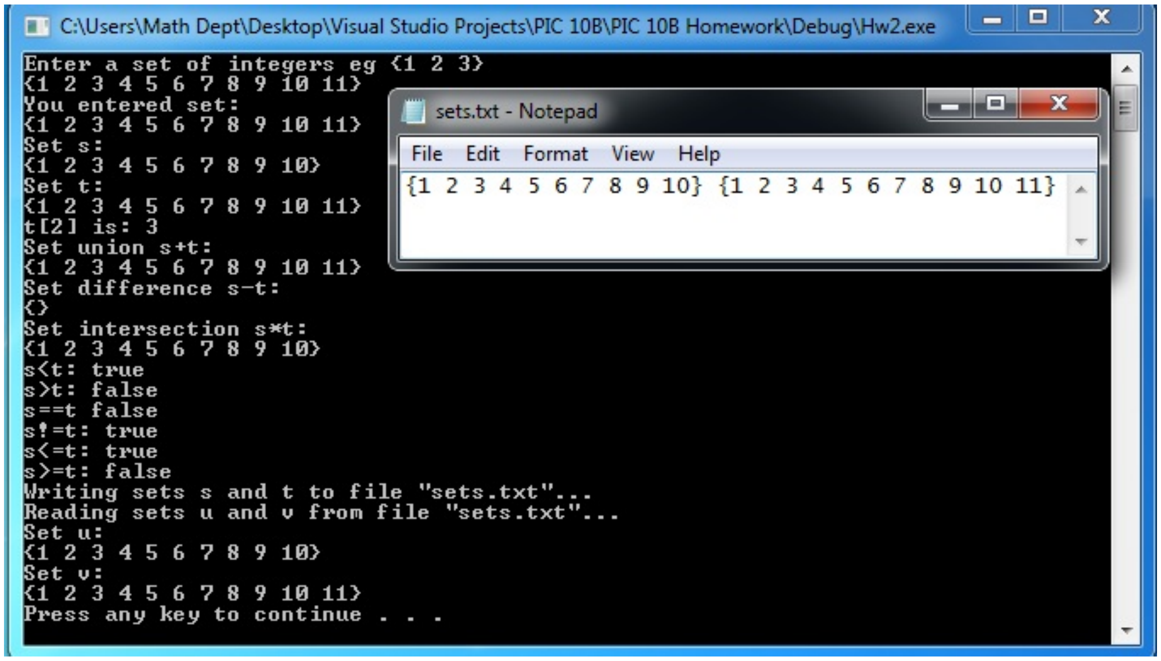
Notice the text file sets.txt is written to and then Set objects are read from it and stored into Set objects u and v.


C:\Users\Math Dept\Desktop\Visual Studio Projects\PIC 10B\PIC 10B Homework\Debug\Hw2....

```
Enter a set of integers eg {1 2 3}
{1 2 3 4 5 6 7 8 9}
You entered set:
{1 2 3 4 5 6 7 8 9}
Set s:
{1 2 3 4 5 6 7 8 9 10}
Set t:
{1 2 3 4 5 6 7 8 9}
t[2] is: 3
Set union s+t:
{1 2 3 4 5 6 7 8 9 10}
Set difference s-t:
{10}
Set intersection s*t:
{1 2 3 4 5 6 7 8 9}
s<t: false
s>t: true
s==t false
s!=t: true
s<=t: false
s>=t: true
Writing sets s and t to file "sets.txt"...
Reading sets u and v from file "sets.txt"...
Set u:
{1 2 3 4 5 6 7 8 9 10}
Set v:
{1 2 3 4 5 6 7 8 9}
```

sets.txt - Notepad

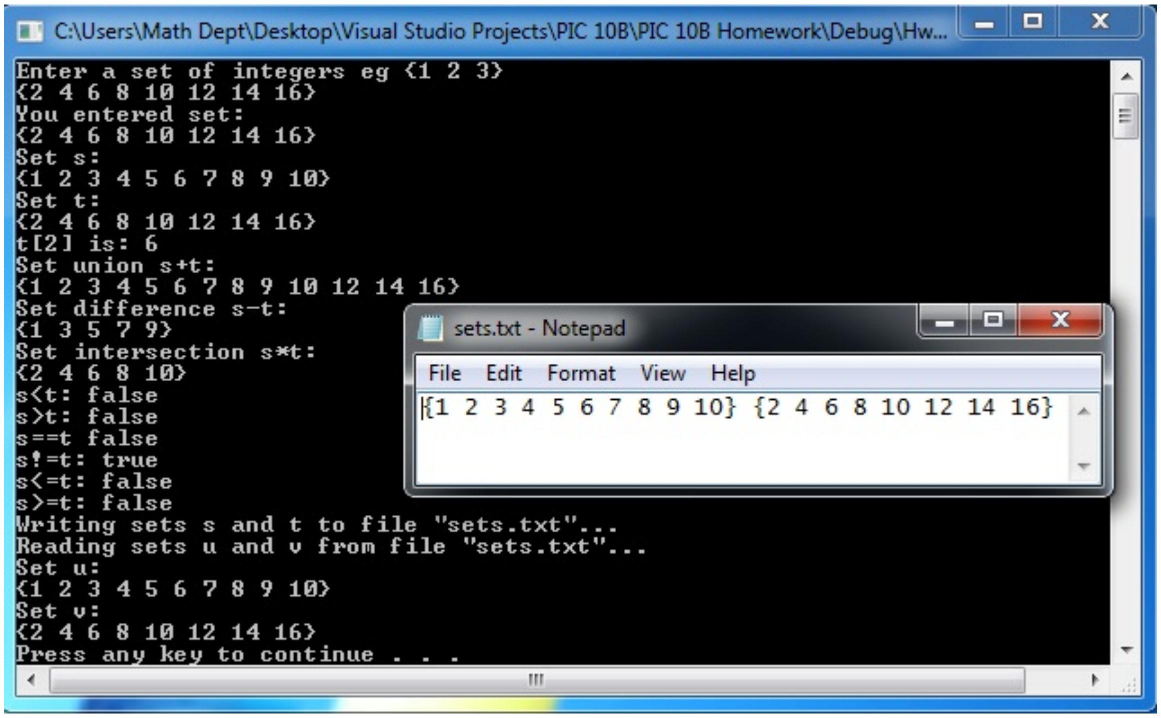
```
{1 2 3 4 5 6 7 8 9 10} {1 2 3 4 5 6 7 8 9}
```


C:\Users\Math Dept\Desktop\Visual Studio Projects\PIC 10B\PIC 10B Homework\Debug\Hw2.exe

```
Enter a set of integers eg {1 2 3}
{1 2 3 4 5 6 7 8 9 10 11}
You entered set:
{1 2 3 4 5 6 7 8 9 10 11}
Set s:
{1 2 3 4 5 6 7 8 9 10}
Set t:
{1 2 3 4 5 6 7 8 9 10 11}
t[2] is: 3
Set union s+t:
{1 2 3 4 5 6 7 8 9 10 11}
Set difference s-t:
{}
Set intersection s*t:
{1 2 3 4 5 6 7 8 9 10}
s<t: true
s>t: false
s==t false
s!=t: true
s<=t: true
s>=t: false
Writing sets s and t to file "sets.txt"...
Reading sets u and v from file "sets.txt"...
Set u:
{1 2 3 4 5 6 7 8 9 10}
Set v:
{1 2 3 4 5 6 7 8 9 10 11}
Press any key to continue . . .
```

sets.txt - Notepad

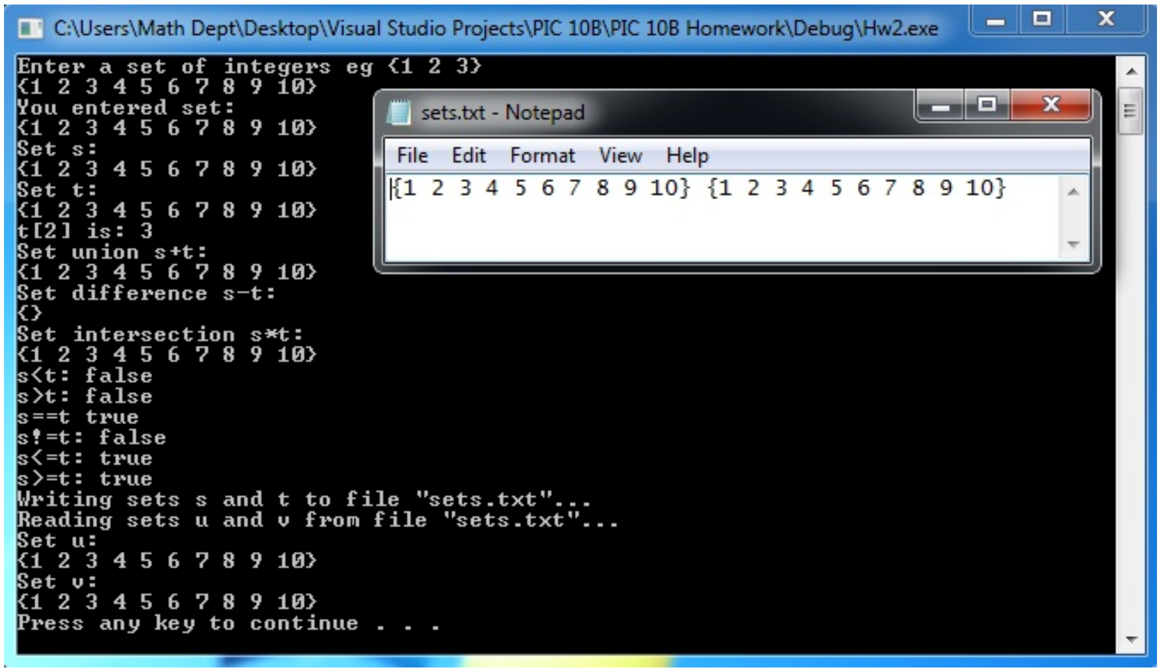
```
{1 2 3 4 5 6 7 8 9 10} {1 2 3 4 5 6 7 8 9 10 11}
```



```
Enter a set of integers eg {1 2 3}
{2 4 6 8 10 12 14 16}
You entered set:
{2 4 6 8 10 12 14 16}
Set s:
{1 2 3 4 5 6 7 8 9 10}
Set t:
{2 4 6 8 10 12 14 16}
t[2] is: 6
Set union s+t:
{1 2 3 4 5 6 7 8 9 10 12 14 16}
Set difference s-t:
{1 3 5 7 9}
Set intersection s*t:
{2 4 6 8 10}
s<t: false
s>t: false
s==t false
s!=t: true
s<=t: false
s>=t: false
Writing sets s and t to file "sets.txt"...
Reading sets u and v from file "sets.txt"...
Set u:
{1 2 3 4 5 6 7 8 9 10}
Set v:
{2 4 6 8 10 12 14 16}
Press any key to continue . . .
```

```
{1 2 3 4 5 6 7 8 9 10} {2 4 6 8 10 12 14 16}
```

C:\Users\Math Dept\Desktop\Visual Studio Projects\PIC 10B\PIC 10B Homework\Debug\Hw...
sets.txt - Notepad
File Edit Format View Help



```
Enter a set of integers eg {1 2 3}
{1 2 3 4 5 6 7 8 9 10}
You entered set:
{1 2 3 4 5 6 7 8 9 10}
Set s:
{1 2 3 4 5 6 7 8 9 10}
Set t:
{1 2 3 4 5 6 7 8 9 10}
t[2] is: 3
Set union s+t:
{1 2 3 4 5 6 7 8 9 10}
Set difference s-t:
{}
Set intersection s*t:
{1 2 3 4 5 6 7 8 9 10}
s<t: false
s>t: false
s==t true
s!=t: false
s<=t: true
s>=t: true
Writing sets s and t to file "sets.txt"...
Reading sets u and v from file "sets.txt"...
Set u:
{1 2 3 4 5 6 7 8 9 10}
Set v:
{1 2 3 4 5 6 7 8 9 10}
Press any key to continue . . .
```

```
{1 2 3 4 5 6 7 8 9 10} {1 2 3 4 5 6 7 8 9 10}
```

C:\Users\Math Dept\Desktop\Visual Studio Projects\PIC 10B\PIC 10B Homework\Debug\Hw2.exe
sets.txt - Notepad
File Edit Format View Help

Grade Breakdown:

Criteria	Description	Points
Header	Starts every .h and .cpp file	1
Comments	Program well-commented.	1
Set	Both constructors defined correctly	2
	contains() and size()	2
	operator[] defined correctly	2
	operator += defined correctly	3
	operator -= defined correctly	3
	operator *= defined correctly	3
	operator <= defined correctly	2
	operators << and >> defined correctly	4
	operators + - * < > == !=	1 pt each
Total		30

A penalty of 5 points will be assessed if your code does not compile using Microsoft Visual Studio 2012.