PIC 10B Lectures 1 and 2 Spring 2015
Homework Assignment #3

Due Thursday, April 23, 2015 by 9:00pm.

Objectives:

1. To redefine the "Big 3": Copy constructor, Assignment operator, and Destructor for a class whose objects manage heap memory.
2. To develop an application that uses function pointers, passes them as arguments to another function, and stores them in an array.

Introduction:

Your supervisor wanted you to make some design changes to your Set class. The biggest change is make your Set objects store their elements not in a vector of `int` values, but on the heap in a dynamic array of `int` values. As a consequence, you will need to redefine the copy constructor, assignment `operator=`, and destructor in order to prevent shallow copies and memory leaks. The specifications for the assignment are below:

The class `Set` will have the following characteristics:

Attributes:
- `elements` (a `int` pointer variable)
- `size` (number of elements in the set)

Constructors:
- `Set()                // creates an empty Set`
- `Set(int element)    // creates a Set with the given element`
- `Set(const Set& s)   // creates a deep copy of Set s`

Each constructor is responsible for initialize member `size`, creating the Set object's dynamic array of values, and having the member `elements` hold the array's pointer. A set with size 0 should have a NULL pointer for `elements`. Otherwise, the size of the array `elements` should always match the value of `size.` Have the copy constructor initialize `elements` to NULL before calling copy to avoid calling `delete` on a wild pointer.

Destructor:
- `~Set()     // deallocates dynamic array elements`

Helper functions (`private` members):

- A procedure `copy(const Set& s)` which makes the calling object a deep copy clone of the given Set s. This function is going to be used by both the copy constructor and the assignment operator (to avoid repeating code). In this function, be sure to avoid self-assignment as well as any memory leaks.
- A procedure `resize(unsigned int new_size)` which does nothing if the `new_size` matches the current `size`. Otherwise, it resizes the array `elements` by having it point to a new dynamic array with `new_size` number of elements. Be sure to avoid any memory leaks. It should only update `size` if `new_size` is strictly smaller than `size`. Also, it should copy as many values of the old array as it can into the new array, leaving any extra elements uninitialized in the case when `new_size` is bigger than `size`.

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 getSize`()` 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=` makes the calling object a deep copy clone of the given Set (using helper function `copy`) and then returns the calling object by reference.
- `operator+=` computes the union of the calling object Set and the Set object passed in and stores the resulting union as the calling object.
- `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.
- `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 general strategy for defining the compound assignment operators *= and -= is as follows: Create an empty local Set object called `answer` to represent the answer. Add elements of the calling object to Set `answer` as needed. Assign the calling object the value of `answer`. Return the calling object. You will have to modify your old definition of operator += to fit the new design.

Stream 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 ostream at the end by reference.
- operator>> should read a Set that is presented in the same format as mentioned above (see operator <<). *This operator will behave slightly differently from the operator >> of your last assignment. It should assign the given Set the value of an empty Set first before doing what it normally does.* Be sure to return the istream at the end by reference.

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.

The application SetApp.cpp will have the following behavior.

1. It will define a function `perform` which returns a Set by value and takes in two Set objects by `const` reference as well as a function pointer parameter called `op`. The function pointer parameter `op` can hold a pointer to any function that takes in two Set objects by `const` reference and that returns a Set by value.
2. It will define function `main()` which will define an array of function pointers whose elements will be the operators `+`, `*`, and `-`. It will also display "Set Calculator" and prompt the user to type in two Set objects using the keyboard, which you will read into two local Set variables. Display the two Sets to the user and then present the user with a menu of options allowing the user to take the union, intersection, or the difference of the two Sets. It should also have another option to perform all three operations in sequence. Each of the first three menu options will have to compute the required Set by calling function perform while passing in the appropriate operator function pointer, displaying the resulting Set to the user. The last option will loop through the array of function pointers and print out each Set obtained by dereferencing the array element and calling the resulting pointee operation.

Directions:

1. Create the project called “Hw3” 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 #3
   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>
*/
```

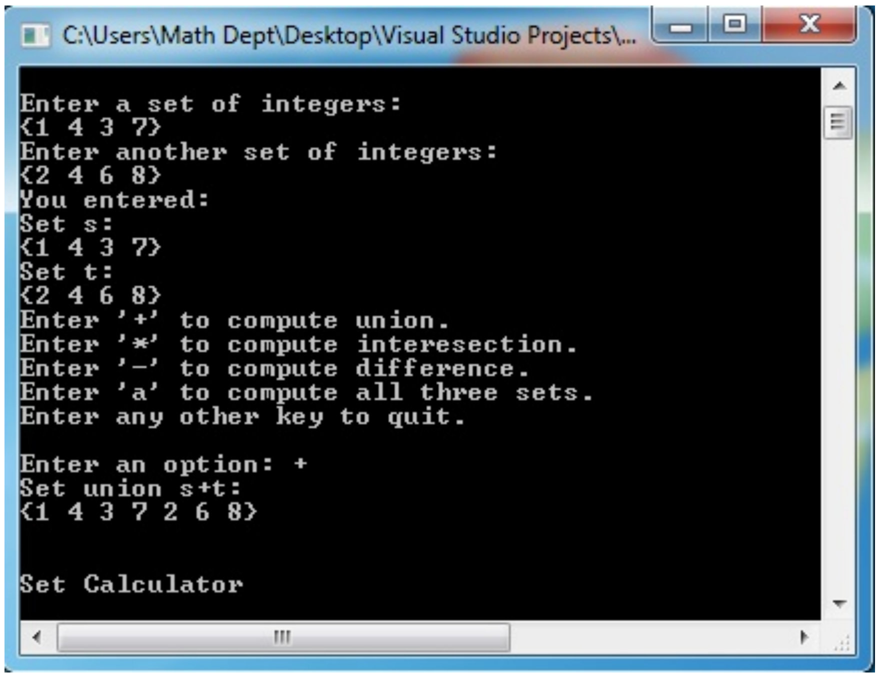
2. Define and implement the Set class as described above. Create a separate interface file Set.h and implementation file Set.cpp for your class. Declare your nonmember operators in Set.h outside the class definition. Define all operators in Set.cpp.

3. Define your application file as specified above and call it SetApp.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
     - SetApp.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.

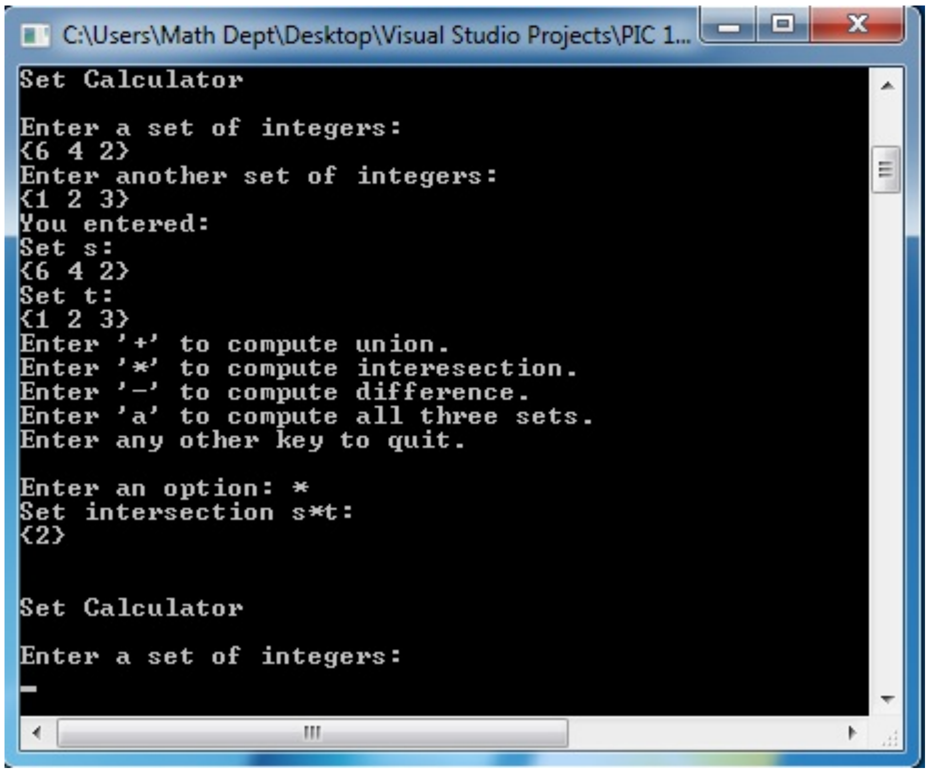
Some screenshots (all the same session) are below:



```
Enter a set of integers:
{1 4 3 7}
Enter another set of integers:
{2 4 6 8}
You entered:
Set s:
{1 4 3 7}
Set t:
{2 4 6 8}
Enter '+' to compute union.
Enter '*' to compute interesection.
Enter '-' to compute difference.
Enter 'a' to compute all three sets.
Enter any other key to quit.

Enter an option: +
Set union s+t:
{1 4 3 7 2 6 8}


Set Calculator
```



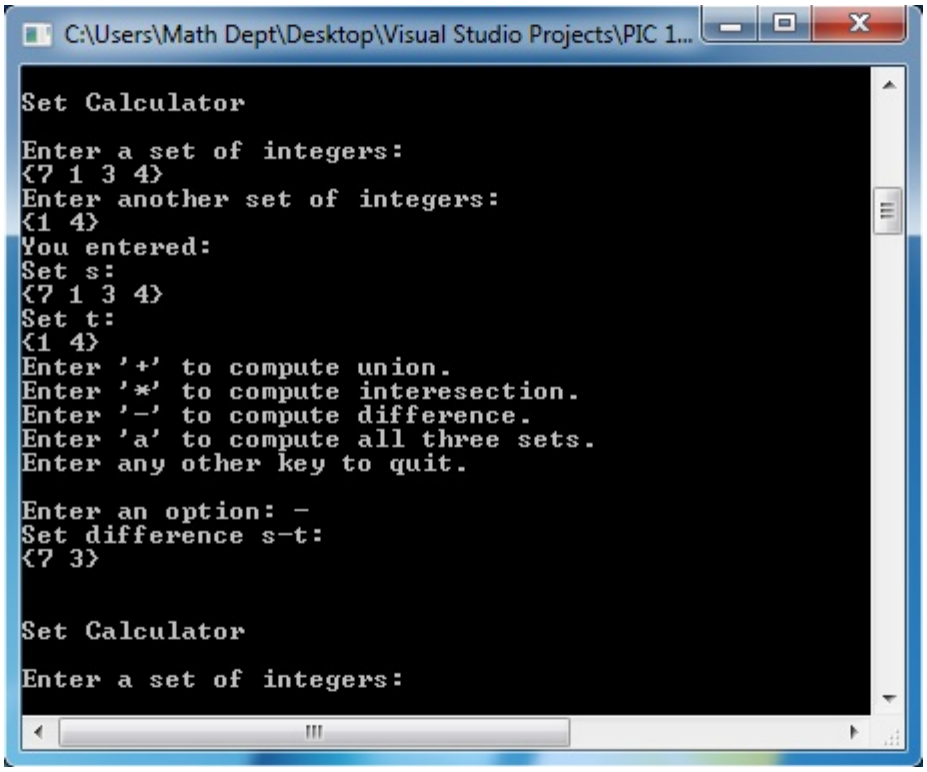
```
Set Calculator

Enter a set of integers:
{6 4 2}
Enter another set of integers:
{1 2 3}
You entered:
Set s:
{6 4 2}
Set t:
{1 2 3}
Enter '+' to compute union.
Enter '*' to compute interesection.
Enter '-' to compute difference.
Enter 'a' to compute all three sets.
Enter any other key to quit.

Enter an option: *
Set intersection s*t:
{2}


Set Calculator

Enter a set of integers:
_
```


C:\Users\Math Dept\Desktop\Visual Studio Projects\PIC 1...

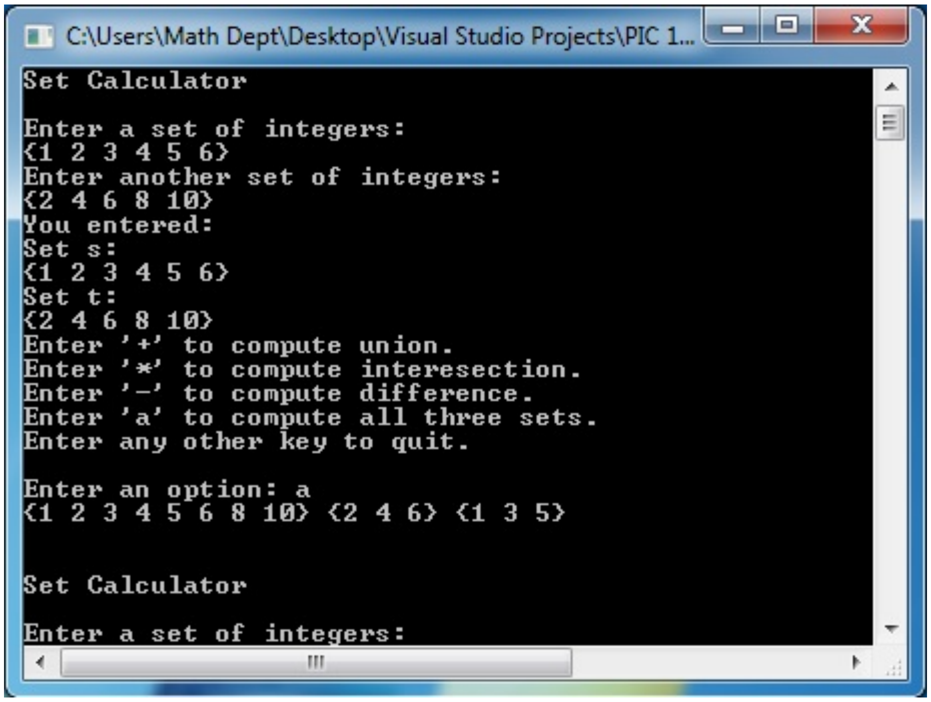
```
Set Calculator

Enter a set of integers:
{7 1 3 4}
Enter another set of integers:
{1 4}
You entered:
Set s:
{7 1 3 4}
Set t:
{1 4}
Enter '+' to compute union.
Enter '*' to compute interesection.
Enter '-' to compute difference.
Enter 'a' to compute all three sets.
Enter any other key to quit.

Enter an option: -
Set difference s-t:
{7 3}


Set Calculator

Enter a set of integers:
```


C:\Users\Math Dept\Desktop\Visual Studio Projects\PIC 1...

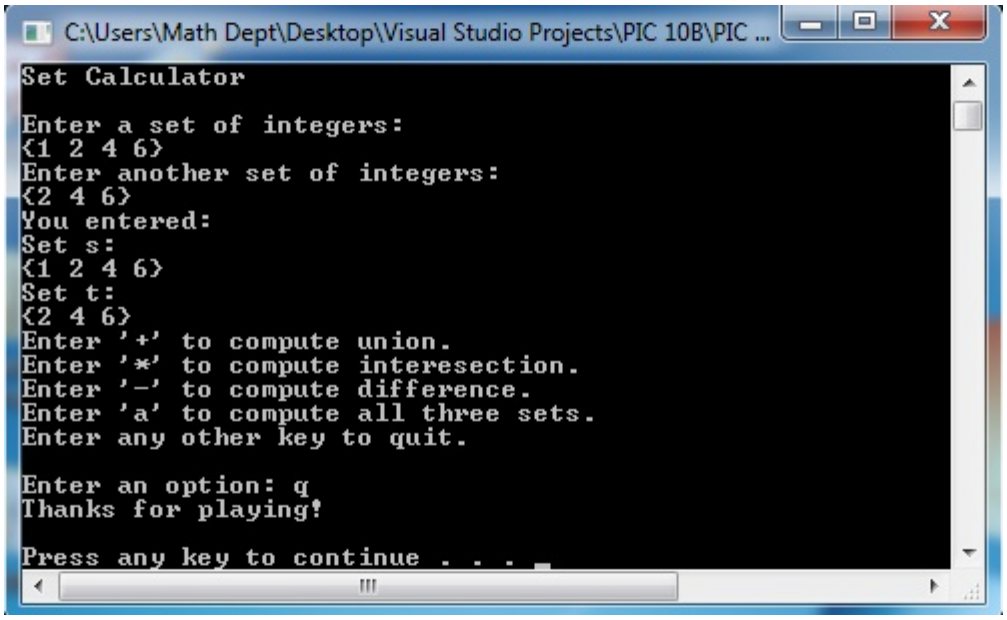
```
Set Calculator

Enter a set of integers:
{1 2 3 4 5 6}
Enter another set of integers:
{2 4 6 8 10}
You entered:
Set s:
{1 2 3 4 5 6}
Set t:
{2 4 6 8 10}
Enter '+' to compute union.
Enter '*' to compute interesection.
Enter '-' to compute difference.
Enter 'a' to compute all three sets.
Enter any other key to quit.

Enter an option: a
{1 2 3 4 5 6 8 10} {2 4 6} {1 3 5}


Set Calculator

Enter a set of integers:
```


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

```
Set Calculator

Enter a set of integers:
{1 2 4 6}
Enter another set of integers:
{2 4 6}
You entered:
Set s:
{1 2 4 6}
Set t:
{2 4 6}
Enter '+' to compute union.
Enter '*' to compute interesection.
Enter '-' to compute difference.
Enter 'a' to compute all three sets.
Enter any other key to quit.

Enter an option: q
Thanks for playing!

Press any key to continue . . .
```

Grade Breakdown:

Criteria	Description	Points
Header	Starts every .h and .cpp file	1
Comments	Program well-commented.	1
Set	Default and conversion constructors correct	2
	Copy constructor	1
	Assignment operator=	1
	Destructor	1
	contains and getSize member functions	1pt each
	copy helper function	3
	resize helper function	3
	operator[] defined correctly	1
	operators +=, -=, and *= defined correctly	1pt each
	operators << and >> defined correctly	3
SetApp.cpp	Function perform	3
	main() defines array of function pointers, inputs two Sets from the keyboard, presents the menu, executes the operation(s) selected as prescribed above. Process repeats until user does not select valid menu option.	6
Total		30

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

You are not allowed to use vector objects anywhere in this solution.