Programming Project 5

EE312 Spring 2013

Due March 7th, 2014 before 5:00 PM CDT
No late submissions accepted!
TEN POINTS
(5 points for Project 4, and 5 more for Project 5)

Special: This is a large project, and we have an exam coming up. To make the project worthwhile, you will be given double time and double points for this project (two weeks, ten points). The first five points will be awarded as "Project 4". Project 4 is just a subset of the requirements below. You have one week to get this subset done. Project 5 is the remainder of the project.

In simpler terms, if you complete key parts of this two-week project early, then you can get up to 5 bonus points (that's what Project 4 really is, bonus points for starting on this project early). If you don't turn in anything by the deadline for Project4, well, you lose the five bonus points, but you can still get full credit (up to 5 points) for Project5, as long as you complete it before Spring break.

The Project4 part of this large project includes insertSet, removeSet, isEqualToSet, isSubset and isMemberSet. You do not need to achieve any particular time complexity bound to get full credit Project4. Specifically, you don't have to use a good algorithm, as long as you don't do something exceedingly convoluted, then you get full credit for Project4. By the time we do the real Project5, you'll not only have to use a good algorithm, you'll also need to know what time complexity (what Big Oh) your algorithm has.

There is a bonus point (i.e., the 5th point) for Project4. To get this bonus point, you must implement all the functions for the Set project. You do not need to achieve any time complexity bounds (i.e., you don't have to have a fast algorithm for any of the functions – just don't do anything exceedingly convoluted). If you write correct implementations for all of the Set functions, *and* you used reasonable coding style, then you should expect to receive five points for Project4. Project5 is about ensuring that you have efficient implementations (which may require you to rewrite several of your Project4 functions to make them faster).

In the instructions below, you will be told to edit "Proiject5.cpp" Since we're splitting the submission into two projects, please edit Project4.cpp first and submit that (in the Project4 directory), then edit Project5.cpp in the Project5 directory next week. Nothing stops you from submitting the same file twice – you can do the whole Project5 in a week. But we do want two separate submissions, one for Project4 and one for Project5.

General: This project will give you experience writing a data structure in C/C++. More importantly, you will get to put some serious effort into using good algorithms in your programs and analyzing their time complexity. Part of your grade for this project will be

based on how efficient your programs are. We will attempt to ascertain if your algorithms require time proportional to N^2, $N \log_2 N$, N, or $\log_2 N$.

Your Mission: Edit the file "Project5.cpp". You should not need to change any other files. You must implement all of the functions defined for the Set ADT (see Set.h for a complete list). I've taken care of several of these functions to get you started. However, the work that remains is **substantial**. Get started early.

General Design: I suggest (actually, I insist) that you keep the elements inside the Set sorted. For example, whatever is stored in elements[0] should be smaller than any other element in the set. Whatever is in elements[1] should be the second smallest, etc.

How Sets Work: Recall that a set is an unordered collection of things. Your Set class will hold a collection of **int**s. The operations you must implement include inserting and removing elements from the set. Please note that the same element **cannot** appear twice in the same set. If *insertSet* is called where the parameter is a number that's already in the set, your function should do nothing. Similarly, *removeSet* should remove an element from the set. If someone attempts to remove an element that's not in the set, then you should do nothing.

In addition to inserting and removing elements, you must provide a membership test. In other words, it must be possible to determine if some arbitrary number is a member in your Set. The *isMemberSet* function should provide this capability. Two sets can also be compared. If the sets have exactly the same elements then they are equal. Note that two sets that each contain zero elements are equal. A set, X is a subset of another set Y if all of the elements in X are also elements of Y. So, a set containing zero elements is a subset of every other set. Also, every set is a subset of itself.

The interesting functions for sets are intersection, union and subtraction. Recall that the intersection of two sets is the elements that are common to both sets. So, if X has the elements {1, 3, 5, 7} and Y has the elements {1, 2, 3, 4} then the intersection of X and Y is a set with elements {1, 3}. The union of two sets is the set of elements that are in either one of the two sets. So, for the same X and Y above, the union would be a set containing {1, 2, 3, 4, 5, 7}. Finally, set subtraction is defined as the set of all elements that are in the first set but are not in the second set. So, if we ask for $X - Y$ we would compute the set containing {5, 7}. You must write all three of these functions.

Performance: For this project we are interested in how efficiently your programs will run. Specifically we're interested in determining for each function you write whether the function takes time proportional to the number of elements in the set, the logarithm of the number of elements in the set, or the square of the number of elements in the set. The best way to determine the efficiency of your algorithm is to analyze the code by hand and count how many operations must be performed. We'll be trying to write a program that will automatically guess the performance. Such programs are very difficult to write, so don't put too much faith in the estimates it gives (it almost always is wrong for isEqualTo). Do your own analysis (besides it will be a great way to study for the next

test). For each of the following functions, the worst-case time complexity must be as described below:

Function	Time Complexity (worst case, N is the number of elements in a set, for operations that apply to two sets, assume both sets have N elements).
insert	O(N)
remove	O(N)
isMember	O(log N)
isEqualTo	O(N)
isSubsetOf	O(N)
unionIn	O(N)
intersectFrom	O(N)
subtractFrom	O(N)