C SC 397a, Spring 2010
Assignment 9, Part 1
Due: **Wednesday**, May 5 at 22:00:00

NOTE: **This is Part 1 of a two-part assignment**. Counting the problems below, a total of 870 assignment points have been issued, 30 points short of the 900 assignment points specified in the syllabus. I'll issue Part 2, covering those last 30 points, no later than 22:00 on Wednesday, April 28.

Problem 1. (90 points) Set

In this problem you are to implement a template class named Set that holds a set of unique values of type T. Here is the interface for Set:

Set()
Creates an empty Set.

Set(const List<T>& L)
Creates a Set from the values in L. If L is empty, the Set is empty.

bool member(T value) const
Returns true if value is a member of the set; false is returned otherwise. Membership is tested via operator==.

void insert(T value)
Inserts value into the set if it is not already a member. If value is already a member, the call has no effect.

void remove(T value)
Removes value from the set, if it is a member. If not, the call has no effect.

int size() const
Returns the number of values currently held by the set.

List<T> values() const
Returns a List<T> of the values currently in the set. The order of values in the list is not defined—any list that contains the values is the set is a correct result of values().

Overload Set<T> + Set<T>, Set<T> * Set<T>, and Set<T> - Set<T> to implement set union, intersection, and difference, respectively. The result of each is a Set<T>, returned by value. Implement these operators as non-member functions that are friends of Set.

Overload Set<T> == Set<T> to test two Sets for equality, yielding a bool. Hint: Use the intersection operation to make this easy.

Your implementation of Set can assume that T has a default constructor, a copy constructor, and support T = T (assignment) , T == T (equality), and iostream insertion (ostream << T). No other assumptions about T may be made. For example, you may not assume that T != T is implemented.

A Set can hold at most 100 values. Any operation that would cause a Set to have more than 100 values results in execution being terminated with an assertion failure.

The List<T> class to be used by Set is in $FILES/a9/a9.h. It is nearly identical to the List class presented in the slides. An additional non-member function is

```
List<string> orderedStrings(const List<T>& L)
```

Note that the result is a List<string>—a list of C++ Standard Library strings (not our String class). The values in the list are the string representations of the values in L, in lexicographic order. The strings are produced by toString(), shown on slide 277 and also in $FILES/a9/a9.h.

Here is an example of using orderedStrings():

```
List<int> L;

L.add(20); L.add(5); L.add(-3);  L.add(2); L.add(10); L.add(1);

cout << orderedStrings(L) << endl;  // Output: [-3, 1, 10, 2, 20, 5]
```

Note that 2 and 20 precede 5 because '2' < '5'. Similarly, -3 precedes 1 because '-' < '1'.

An inserter for Set is provided in $FILES/a9/SetIns.h. It uses Set::values() and orderedStrings().

Here is an example of Set in operation: ($FILES/a9/s0.cc)

```
#include "Set.h"

int main()
{
   Set<char> s1;
   for (const char *p = "testing"; *p; p++)
      s1.insert(*p);

   List<char> s1vals = s1.values();
   cout << s1vals << endl;        // Output: [t, e, s, i, n, g] (Order is NOT defined!!)

   cout << s1 << endl;            // Output: {e, g, i, n, s, t} (Uses inserter in a9.h)

   Set<char> vowels;

   for (const char *p = "aeiou"; *p; p++)
      vowels.insert(*p);

   cout << vowels << endl;        // Output: {a, e, i, o, u}

   Set<char> u = s1 + vowels;
   cout << u << endl;             // Output: {a, e, g, i, n, o, s, t, u}

   Set<char> i = s1 * vowels;
   cout << i << endl;             // Output: {e, i}

   Set<char> d = s1 - vowels;
   cout << d << endl;             // Output: {g, n, s, t}
}
```

Implementation Notes

To avoid some tedious issues involving template instantiation simply put all your code for Set in one file: Set.h. A starter implementation can be found in $FILES/a9/Set.h. Note that it #includes a9.h and SetIns.h.

For grading your code will be compiled with the test programs like this:

```
g++ -o s0 -I$FILES/a9 s0.cc
```

The -I option indicates that the specified directory is to be searched for include files. Be sure that your Set.h works

with this exact g++ invocation.

Make your Set implementation as simple as possible—don't turn this into a data structures problem and don't worry about efficiency! For example, a representation with T itsValues[100]; and a count of values is fine. However, if you want to learn a little more, use the C++ Standard Library vector class to hold the values in your set.

Use the C++ Standard Library string class, not our String class.

RESTRICTION: using the library's set, multiset, map, and multimap classes is prohibited.

Problem 2. (25 points) vowels.cc

Write a program that reads lines from standard input and for each line, replaces vowels with underscores and prints the number of vowels found, followed by the altered line. Example:

```
% cat in
Bjarne
Stroustrup
% vowels < in
  2 removed: Bj_rn_
  3 removed: Str__str_p
%
```

Note that the vowel count is right-justified in a three-character field and followed by a blank. (Don't worry about overflowing the field.)

RESTRICTION: The only instance of a control structure that may appear in your solution is a while loop to read lines from standard input. (See slide 296 for an example.) Approach the problem by creating a function object (slide 297) that tests whether a character is a vowel ("aeiou", upper- or lower-case) and use that in conjunction with the count_if and replace_if STL algorithms.

The count_if algorithm is shown on slide 301. replace_if is very similar:

replace_if(Iterator first, Iterator last, Predicate pred, Value replacement)

Use the C++ Standard Library string class, not our String class.

You'll need to #include <algorithm> to use count_if() and replace_if().

Regarding the restriction on control structures, aside from that single while loop to read lines, you may not use while, do-while, for, if, switch, or the ternary conditional operator, ?:.

Problem 3. (16 points) inherit.txt

In this problem you are to find and describe a real-world example of multiple inheritance. You must cite two base classes and a class that is derived from both. Both base classes must contain at least one data member and have at least one member function. Present your example in the style of the ClockRadio example, on the first of the multiple inheritance slides.

To encourage creativity, your grade will be based on a subjective judgement of how creative your example is. An example that is "creative" will receive the full 16 points. A "somewhat creative" solution will be worth 12 points. A "not very creative" solution will be worth 8 points. Note that the bar for "creative" is not very high: ClockRadio would be worth 16 points if it weren't already an example in the slides! A "not very creative" example of single inheritance is the well-worn Shape/Circle/Rectangle hierarchy.

You may work freely with any number of your classmates on this problem but each student in a collaborative group must turnin inherit.txt and list the members of the group in that file.

Problem 4. (Extra Credit) extra.txt

Submit a plain text file named extra.txt with the following.

(a) (1 point extra credit) Estimate how long it took you to complete this assignment. Other comments about the assignment are welcome, too. I appreciate all feedback, favorable or not.

(b) (1-3 points extra credit) Cite an interesting course-related observation (or observations) that you made while working on the assignment. The observation should have at least a little bit of depth. Think of me saying "Good!" as one point, "Interesting!" as two points, and "Wow!" as three points. I'm looking for quality, not quantity.

Miscellaneous

You must make good use of C++, including but not limited to:
Using bool, const, and member initializers when appropriate
Using new and delete rather than malloc and free
No public data members
No memory leaks

Test programs for Set with names of the form s?.cc and corresponding executable reference versions are in $FILES/a9. Each of the test programs has a comment designating the point value. Each is all or nothing for the specified point value; if there's a diff, even one byte, it'll be zero points for that one. There's only one input file, vowels.in, for vowels.cc. It's worth all 25 points. Use diff to be sure that the output of your solutions exactly matches the output of the reference versions! The set of test programs, reference versions, and any associated data files will freeze exactly one week (168 hours) prior to the due date/time.

$FILES/a9/{testset,testvowels} are simple test scripts.

Feel free to use comments to document your source code as you see fit, but note that no comments are required.

You should be able to complete problems 1 and 2 using the material on slides 1-303.

Don't hesitate to ask the instructor for hints and/or help if you have trouble with a problem.

Deliverables

Use `turnin` with the tag 397a_9 to submit your solutions for grading. The deliverables for this part of the assignment are Set.h, vowels.cc, inherit.txt, and if you choose to submit it, extra.txt. (Again, note that there is to be no Set.cc—all the code goes in Set.h.)