CS107 Assignment 3: Frequently Asked Questions

1. **Question:** I see a sample application called **thesaurus-lookup**, but I didn't see any mention of that in the handout. Are we expected to do anything for this?

   Answer: No. The only changes you need to make are to **vector.[hc]** and **hashset.[hc]**. You can test your implementations of your vector and your hashset by running vector-test and hashset-test to ensure that the output makes sense. When you're sure that you're done coding, you can use the supplied **streamtokenizer.[hc]** and **thesaurus-lookup.c** files to build another, more sophisticated test application that exercises the generic C containers to do more meaningful things. It's a good thing to play with, because you'll be writing an even more sophisticated application for Assignment 4, which has you use your vector, your hashset, this streaktokenizer type, and a few more custom types. In particular, look at the **thesaurus-lookup.c** file to get some clues on how you store C strings in a generic container like the vector and the hashset. It'll be helpful come Assignment 4.

2. **Question:** The specifications for this question says that if the vector is not sorted, we are to use a linear search algorithm. Should we build that lsearch function, or is there a pre-made lsearch function? I looked for lsearch and could not find it in the standard c libraries.

   Answer: There is an **lsearch** function that's part of the gcc version of the C library. You can find the reference on an Elaine workstation using the command "man -s 3c lsearch". If you just type "man lsearch" it gives you the reference for an lsearch command line program, which isn't what you're looking for. That being said, you'll probably want to use **lfind** instead of **lsearch**. Read the documentation that comes up and you'll understand why.

3. **Question:** Is it a mistake to allow VectorInsert to insert off the end of the array?

   Answer: No, it's actually right. When an array has 10 elements, the legal insertion points are 0-10. To insert at position 10 is just to tack the new element onto the end.

4. **Question:** In the vector class interface, some of the functions say that "an assert is raised if ...". What is an assert and how do I use it?

   Answer: Assert is a very handy (mostly debugging) runtime macro. Asserts are statements you add to your code to verify that assumptions you make in your code are actually true. If not, when you run your program, it will fail on the assert statement, reporting which line failed.

   The standard assert macro can be used to detect and report exceptional conditions. This macro takes one argument, an expression which is evaluated and if true (non-zero) execution continues on, but if false (zero) results, it will print an error and terminate the program. The idea is to assert what must be true before carrying out an operation that depends on those assumptions.

```
       #include <assert.h> // need to include this to see macro
definition
       {
          int i, scores[MAX_SCORES];
          i = SomethingComplicated(); // i should be in bounds,
but is it really?
          assert(i >= 0 && i < MAX_SCORES); // assert before we
access
          scores[i] = ...
```

   In industry often the "production" version of the code will **not** include the assert statements, so never put an expression that changes the state or data inside an assert statement. Here is an example of **what not to do:**

```
        #include <assert.h>
 {
    int i, scores[MAX_SCORES];
```

```
    assert(i = 5);  // will return TRUE but you are assigning i
inside
                    // the assert. This is BAD!
    scores[i] = ...
```

To find out more about assert, read its man page.

5. **Question:** Whenever I have errors during compilation, the variables in my error messages are displayed as 'â' instead of the actual variable name.

   Answer: Add the following line to the file "~/.cshrc":
   setenv LC_CTYPE C