CS 7A - Spring 2015 - Initializing and concatenating string objects. Due 2/3/15

In Chapter 3 of PPP (Programming Principles and Practices) we learn about objects, input, variables, type, operations and operators, assignment and initialization, and type safety.

1. Pick three definitions from chapter 3 ("object," "type," "definition," "value," "variable," "declaration," and "type safety") and write a program will declare and initialize string variables and print those to the console.

   Some definitions are short enough to fit on one line, like

   > **bug:** an error in a program.

   ...while others don't, like

   > **correctness:** a program or a piece of a program is correct if it meets its specification. Unfortunately, a specification can be incomplete or inconsistent, or can fail to meet users reasonable expectations. Thus, to produce acceptable code, we sometimes have to do more than just follow the formal specification.

   When declaring and initializing a `string`, you can't add the string literals when initializing the string, as in

   ```
   string bug = "an " + "error " + "in " + "a " + "program."; // error!
   ```

   Why is that? A good way to get explanations for things like this is to visit helpful internet sites such as stackoverflow.com (try following that link, for instance, and then this onemultiline-string-literal

   You also can't have a carriage return in the middle of a string literal, like this

   ```
   string bug = "an error in
   a program."; // error
   ```

   What to do? In class we showed how to create a sequence of `string` variables and then add (concatenate) them like so:

   ```
   string bug01 = "an error in ";
   string bug02 = "a program.";
   string bug = bug01 + bug02; // this works!
   ```

   Or, as the second stackoverflow.com link describes, is you just put the string literals adjacent (as shown in the example below) the compiler will automatically concatenate them. Pretty nifty!

```
1  // Geoff Hagopian
   // Printing definitions with concatenated strings.
3
   #include <iostream>
5  using namespace std;

7  int main() {
       string correctness_def =
9          "correctness: a program or a piece of a program is correct if it"
           " meets its specification. Unfortunately, a specification can be"
11         " incomplete or inconsistent, or can fail to meet users' reasonable"
           " expectations. Thus, to produce acceptable code, we sometimes have"
13         " to do more than just follow the formal specification.";
       cout << correctness_def;
15 }
```