CS 111 - Week 13 Lab Exercise - 2016-11-17

Deadline

Due by the end of lab. (Submit whatever you have by the end of lab, even if incomplete.)

How to submit

Submit your resulting `.cpp` and `.h` files for this lab using `~st10/111submit` on nrs-labs, with a homework number of **93**.

Purpose

To practice writing functions containing count-controlled `while` loops, writing `main` functions that test them, and writing an interactive front end for one of them.

Important notes

- You are required to work in **pairs** on this lab exercise. If you are not pair-programming, then you may not receive full credit for your lab exercise.
- Put **both of your names** either at the beginning or the end of your `purpose` statements for lab exercise functions.
- If you have a question during lab, and I am helping another pair, add one or both of your names to the "Next:" list on the board, and I will get to you as soon as I can.

Problem 1

Problem 1 part a

For some practice writing a count-controlled loop:

Using the design recipe, design and write a function `wordblock` that expects a string, and returns the length of that string, but also has the side-effect of printing to the screen that string the same number of times as its length, each on its own line -- that is,

```
wordblock("moo") == 3
```

...and has the side-effect of printing to the screen:

```
moo
moo
moo
```

And,

```
wordblock("dinosaur") == 8
```

...and has the side-effect of printing to the screen:

```
dinosaur
```

```
dinosaur
dinosaur
dinosaur
dinosaur
dinosaur
dinosaur
dinosaur
```

Problem 1 part b

Now, to formally test `wordblock`, design a `main` function in a file named `wordblock_test.cpp` that:

- prints a message saying that you are testing function `wordblock`
- puts `boolalpha` into the `cout` output stream, so that `bool` values are printed as `true` and `false`
- because `wordblock` has a desired side-effect in addition to its return value, we need to be more specific about what should be seen as a result of its test calls -- so,

  for EACH of `wordblock`'s examples/tests,

  - it should **first** print a message saying that what follows should be the string <desired_string> repeated <its length> times, followed by `true`,
  - and **then** put that example/test in its own separate `cout` statement, such that the result of that test will be printed on its own line

You can compile your `wordblock_test.cpp` with:

```
g++ wordblock_test.cpp wordblock.cpp -o wordblock_test
```

Problem 2

Problem 2 part a

For some more practice writing a count-controlled loop:

Using the design recipe, design and write a function `starline` that expects the desired number of asterisks/stars to print to the screen, and it returns the number of asterisks/stars printed to the screen, but also has the side-effect of actually printing to the screen that many asterisks on a **single** line, followed by a newline character. (If an integer that is <= 0 is given, **no** asterisks should be printed, and it should return 0, since no asterisks were printed.)

Problem 2 part b

Now, to formally test `starline`, design a `main` function in a file named `starline_test.cpp` that:

- prints a message saying that you are testing function `starline`
- puts `boolalpha` into the `cout` output stream, so that `bool` values are printed as `true` and `false`
- because `starline` has a desired side-effect in addition to its return value, we need to be more specific about what should be seen as a result of its test calls -- so,

  for EACH of `starline`'s examples/tests,

- it should **first** print a message saying that what follows should be a line containing <desired number> asterisks, followed by `true`,
- and **then** put that example/test in its own separate `cout` statement, such that the result of that test will be printed on its own line

You can compile your `starline_test.cpp` with:

```
g++ starline_test.cpp starline.cpp -o starline_test
```

Problem 3

Decide: which of `wordblock` or `starline` would you like to to have an interactive front end for?

Create a `main` function in a file named EITHER `starline_ask.cpp` OR `wordblock_ask.cpp` that provides an interactive front end for `starline` or `wordblock`, respectively (and gives you a little bit more practice using `cin`, along with some more practice with another local variable):

- it should interactively ask the user either how many stars or what repeated-word they want to see,
- and then it calls `starline` or `wordblock` with the value entered by the user.

You can compile your `starline_ask.cpp` with:

```
g++ starline_ask.cpp starline.cpp -o starline_ask
```

...or your `wordblock_ask.cpp` with:

```
g++ wordblock_ask.cpp wordblock.cpp -o wordblock_ask
```

Remember:

Using `~st10/111submit` on `nrs-labs`, submit the resulting `.cpp` and `.h` files, using a lab number of **93**.