CS 161 Lab #3 – Design and Loops

- Start by getting checked off for (up to) 10 points of work from the previous lab in the first 10 minutes.
- To get credit for this lab, you must be checked off by a TA by the end of lab.

Goals:

- Expand your list of Vim commands
- Design a solution before implementation
- Identify test cases before implementation
- Practice using loops

(2 pts) A. More Adventures with Vim

First, create a **lab3** directory in your **labs** directory, and change into the **lab3** directory.

```
$ cd labs
$ mkdir lab3
$ cd lab3
```

Now, let's practice using the vim editor following the instructions below.
(If needed, refer to Lab #1 for a reference guide to the basic commands)

1. Before you start, ensure that you copied this .vimrc file to your home directory (**~**), from Lab 2. It specifies some nice vim configurations, like auto-indenting code as you type it. The changes take effect the next time you run vim.
   ```
   $ cp ~wagstafk/cs161/files/.vimrc ~
   ```
2. Open a test file for practice:
   ```
   $ lab3_vim.cpp
   ```
3. Save the file using **:w** <Press Return>
4. Go into insert mode by pressing **i**
5. Type a standard starting program in as follows:
   ```
   #include <iostream>
   using namespace std;
   int main() {
     return 0;
   }
   ```
6. Switch from insert mode to command mode by pressing <Esc>.
7. Go to the third line (or whichever line has `main()`) in your code by typing **:3**
8. **Delete** this line of code by typing **dd**
9. Put the line of code back underneath the first line by moving your cursor to the top line and typing **p** for **paste**.
10. Type **u** to **undo** the paste.
11. Go back to where the line should be, and **paste** it using **p**
12. Now, **copy** the line of code by typing **yy** (If you want to copy more than 1 line, you type one less than the number of lines you want to copy after the **y**, e.g., **y4** copies 5 lines).
13. Paste the line of code by moving your cursor to a line and typing **p**
14. Undo the paste by typing **u**
15. Try using the **j, k, h**, and **l** keys to move around the screen. You can also use the arrow keys, but these don't always work with all possible terminals/connections.

16. Search for “main” in vim with **/main**
17. *Switch from command mode to insert mode,* **i**, and type some random letters, such as **dkfjdsl**, then press <Esc> to *switch back to command mode*.
18. Use **x** to delete (one at a time) all the characters from the random string you typed.
19. Use **u** to undo anything at any time!
20. Use **<Control>+r** to re-do what you un-did.
21. Practice switching between vim and the shell as follows:
    a. From command mode in vim, press **<Control>+z**
    b. Now you are in the shell and your vim editor is running in the "background."
    c. You can navigate through your files, compile programs, run programs, etc.
    d. When you want to return to vim for more editing, type
       `$ fg`
       This will return vim to the "foreground" so you can do more editing.
    e. Switch back to the shell with <Control>+z, then back to vim with **`fg`**, until you are comfortable going back and forth.
    f. Now you can alternate between editing your program and compiling/testing it without having to stop and start vim each time.
22. When you are done, save and quit vim with **:wq**

Here are some Linux and vim cheat sheets you can use as a quick reference:
https://canvas.oregonstate.edu/courses/1770357/pages/references-linux-vim-c++-etc-dot

---

(3 pts) B. The Repeating Message

Your TAs will introduce examples of for, while, and do-while loops. Pay close attention and ask questions about any details that are confusing.

Next, **you will work with other student(s) in a small group.** Go through the steps together and help each other figure things out! You should each complete all steps. Ask for a TA if you are stuck.

1. Create a **labs/lab3** directory and use vim to start a new program called **lab3.cpp** .
2. Write a **for loop** to print a message of your choice the number of times the user chooses. You will need to prompt the user for the number of repetitions (store it in an **int**) before the for loop begins.
   Example run of the program (user input is highlighted):

```
How many times would you like to see my message? 3

He who learns but does not think, is lost! -- Confucius
He who learns but does not think, is lost! -- Confucius
He who learns but does not think, is lost! -- Confucius
```

3. Your program should print the message 0 times if the user enters 0. Test that this works.
4. Add a check for whether the user entered a negative number. Negative numbers don't make sense, so in this case the program should print an error and exit (using return 1).
5. Test that your program behaves as expected if the user enters -3.
6. Test that your program (still) behaves as expected if the user enters a positive number or 0.

(5 pts) C. Mirror Writing

Your goal is to write a program that reads in the user's name and writes it out backwards.

Example runs of the program (user input is highlighted):

```
Please enter your name: Alfred
derflA

Please enter your name: Anna
annA

Please enter your name: Supercalifragilisticexpialidocious
suoicodilaipxecitsiligarfilacrepuS
```

(2 pts) Design

Before starting to write your program, first take some time to discuss it with your partners and plan how it will work.

The point of design is to give you a plan to follow while you are coding. This saves time later debugging your program because you can catch mistakes early (sometimes before any code is written). It is better to spend one hour of designing than it is to spend five hours debugging.

You will implement your solution only after finishing the design.

(1 pt) Draw a **flowchart** to show how this program will execute. Use rectangles for command/statements and diamonds for decision points. Use arrows to connect elements in the flowchart and illustrate how loops work.

You can draw on **paper** (and scan/take a picture) or use **PowerPoint** or other drawing program, or use this free website: https://app.diagrams.net/

(1 pt) Write down (somewhere you can show your TA - Word, Google doc, etc.) at least 6 test cases that include at least 1 good, 1 bad, and 1 edge case. The first row shows an example.

Test setting	User input	Expected result
`Please enter your name:`	`Anna`	`annA`

Your design (flowchart and test cases) must be checked off by a TA before proceeding to implementation.

(3 pts) Implementation

Choose the kind of loop you want to use (for loop, while loop, do-while loop).
Consider what variables you will need to store information.
Implement your mirror writing in **lab3.cpp (it's fine if this program contains both the repeating message and the mirror writing).**

Tips:

- It is fine to use an "int" variable to be the loop counter.
- If you have a string called `day`, then `day.length()` is the length of the string.
- You can access individual characters inside a string with `day[3]` for character 3 or `day[i]` for character i.
- Indexing starts at 0. Therefore, if `day` = "today" and i = `day.length()`, then `day[i]` does not exist, because `day.length()` = 5 and valid indices for `day` go from 0 to 4. Adjust your loop starting/stopping conditions to fit.
- You can make a loop count down by using the decrement operator (`i--`).

D. Get your work checked off by a TA.

Submit your program (lab3.cpp) on Canvas (this can be done while you are waiting to be checked off).

All partners' names must be in the comment header to get credit.

1. Transfer your .cpp files from the ENGR servers to your local laptop.
2. Go to the Lab 3 assignment: https://oregonstate.instructure.com/courses/1770357/assignments/7847924
3. Click "Submit Assignment".
4. Upload your .cpp file (lab3.cpp).
5. Click the **"Submit Assignment"** button.
6. You are done!

If you finish the lab early, this is a golden chance to continue working on Assignment 2 (with TAs nearby to answer questions!).
Remember, the assignment you submit must be your work alone (no partners).
