CS 161 Lab #9 – Recursion

- Get checked off for **(up to) 3 points** of work from the previous lab in the first 10 minutes, **if you had a non-zero grade for Lab 8 already**.
- **To get credit for this lab, you must be checked off by a TA by the end of lab.**
- **The implementation part of this lab (not the quiz) can use pair programming.**

Goals:

- Develop recursive functions to solve problems
- Compare iterative and recursive solutions to the same problem

---

(3 pts) A. Lab Quiz (Canvas)

Visit this link on Canvas to take the Lab 9 quiz:
https://oregonstate.instructure.com/courses/1771939/quizzes/2536378
Re-take the quiz until you get all of the questions right! Canvas saves your last score, not your highest score. If you don't get 100% within the time available, finish outside of lab.

(5 pts) B. Recursion vs. iteration

We've talked about some pros and cons of iteration versus recursion. One consideration is how long it takes the program to run. Download the following starter code (using `wget`) which will allow you to measure the runtime of iterative versus recursive solutions to the following problem.
http://classes.engr.oregonstate.edu/eecs/winter2020/cs161-020/labs/lab9_recur.cpp

For this program, we will generate Fibonacci numbers. Each Fibonacci number is the sum of the two preceding numbers, with starting definitions of $F_0 = 0$ and $F_1 = 1$.

- Thus: F_0 is 0, F_1 is 1, F_2 is 1, F_3 is 2, F_4 is 3, F_5 is 5, F_6 is 8, etc.
- Written with a **recursive definition**: $F_i = F_{i-1} + F_{i-2}$ for $i > 1$, where $F_0 = 0$ and $F_1 = 1$
- This definition has **two base cases**: F_0 and F_1

The starter file includes an iterative function that returns the *n*th Fibonacci number, F_n **.**

(1 pt) Add a file header as usual with your name (and partner's name if pair programming), date, description, etc. Then add code to read *n* from the user and then call `fib_iter(n)`. Run it for several test cases (values of *n*).

(2 pts) Next, write a recursive function called `fib_recur()` that also takes one parameter, *n*, of type `int`, and returns the *n*th Fibonacci number, F_n. Run it for several test cases to confirm it gives the same result as the iterative version.

```
int fib_recur(int n);
```

(1 pt) Time the iterative and recursive solutions for finding the 1st, 5th, 15th, 25th, and 45th Fibonacci numbers. Compare and comment on your results.

(1 pt) Write down and show your TA a trace through the recursive solution for n=5. That is, show at each step what this expands to in terms of new function calls:

fib_recur(5) = fib_recur(4) + fib_recur(3)
= …. <you fill in the rest>

(1 pt) Stair climbing: You're standing at the base of a staircase and are heading to the top. A small stride will move up one stair, while a large stride advances two stairs. Let $\mathbf{C_n}$ be the number of ways to climb *n* stairs using some combination of large and small strides. For example, a staircase of 3 steps can be climbed in 3 different ways:

3 small strides, 1 small + 1 large, or 1 large + 1 small.

- Can you express $\mathbf{C_n}$ using a recursive definition (depends on $\mathbf{C_i}$ for some *i* value(s) less than *n*)?
    - What are the base cases for counting the ways you can climb stairs by going one stair or two stairs at a time?
    - What is the recursive step?
        - Hint: To climb *n* stairs, the final stride is either a small stride or a large stride. If it is a small stride, what can you use from some smaller *i*? If it is a large stride, what can you use from some smaller *i*? Then, how would you put these together to get the number of ways you can climb *n* stairs?
    - How many different ways can you climb 4 stairs? 5 stairs?
    - Is there a connection between this sequence and what you found with Fibonacci numbers?

(1 pt) Implement this solution as a recursive function that takes in the number of stairs *s*:

```
int n_ways_climb(int s);
```

Run it for several test cases (values of *s*).

(2 pts) C. Recursion with strings

Recursion is useful in other situations, not just for computing mathematical sequences. For this part, your goal is to write a recursive function that create palindromic strings based on an input character. A palindromic string reads the same forwards and backwards.

- Create a new file, `lab9_recur_strings.cpp` in which you will implement a function that takes in a character and returns a C++ string:
  `string pal(char c);`
- Look for the pattern in these examples of the input character and the resulting string:
    - `pal('A') => "A"`
    - `pal('B') => "ABA"`
    - `pal('C') => "ABACABA"`
    - `pal('D') => "ABACABADABACABA"`
- Design:
    - What is the base case?
    - What is the recursive step?
      (how you can you write `pal(c)` in terms of `pal(c-1)`?)
        - Tip: You may need to refer to `pal(c-1)` more than once.
- **(1 pt) Get your design checked off by a TA before proceeding to implementation.**

- **(1 pt)** Implement your function. You can assume that the input character is a capital letter from A to Z. Call it multiple times from `main()` to confirm that it works for the test cases above (and any others you want to test).

Ask a TA to check your work (get points) and then submit your programs to **TEACH**

1. Transfer your .cpp files from the ENGR servers to your local laptop.
2. Connect to TEACH here: https://teach.engr.oregonstate.edu/teach.php
3. In the menu on the right side, go to **Class Tools -> Submit Assignment.**
4. Select **CS 161 020 Lab_9** from the list of assignments and click "SUBMIT NOW".
5. Select your .cpp file (`lab9_recur.cpp, lab9_recur_strings.cpp`).
6. Click the **Submit** button.
7. You are done!

Point totals: 3 pts (quiz) + 5 pts (iteration vs. recursion) + 2 pts (recursion with strings)

If you finish the lab early, this is a chance to work on your Assignment 5 implementation (with TAs nearby to answer questions!).
