CS 161 Lab #9 – Recursion

- Start by getting checked off for (up to) 3 points of work from Lab 8 in the first 10 minutes.
- Your goal is to get all points for Lab 9 checked off by a TA by the end of this lab, so that you do not need to do additional work outside of the lab.
- You will work in small groups to complete the programming assignments, with each person writing their own program. Include each partner's name in your file header.

Goals:

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

---

(5 pts) A. 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. Here you will get a chance to compare how long it takes to run iterative and recursive solutions to the same problem.

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

(1 pt) Here is an example of a trace through the recursive solution for n=3.

```
fib_recur(3) = fib_recur(2) + fib_recur(1)
             = (fib_recur(1) + fib_recur(0)) + 1
             = (1 + 0) + 1
             = 1 + 1
             = 2
```

(Check against the list above. Yep, F_3 is 2!)

Write down and show a TA your 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) Create a `lab9` directory, cd in to it, and copy the following starter code to your directory that will allow you to measure the runtime of iterative versus recursive solutions (don't forget the final period).

```
$ cp ~wagstafk/cs161/files/lab9_recur.cpp .
```

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

Add a file header as usual with your name and your partners' names, 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, implement the 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.

Your function should have this format:

```
int fib_recur(int n) {
    if ( . . . ) { /* base case 1 (F0) */
        return something;
    } else if ( . . . ) { /* base case 2 (F1) */
        return something;
    } else { /* recursive call (see recursive definition of Fi above) */
        return something that uses fib_recur(n-1) and fib_recur(n-2);
    }
}
```

(1 pt) Time the iterative and recursive solutions for finding the 1st, 5th, 15th, 25th, and 45th Fibonacci numbers. Is it what you expected? Compare your results with your partners.

(5 pts) B. Count digits

Next let's experiment with recursion to count the digits in an integer.

You will implement a function that takes in an integer and returns the number of digits:
`int n_digits(int n);`

Examples (test cases)

- `n_digits(3) => 1`
- `n_digits(53) => 2`
- `n_digits(8675309) => 7`

(1 pt) Design:

- What is the base case?
- What is the recursive step?
  (how you can you write `n_digits(n)` in terms of `n_digits(n/10)`?)

Get your design checked off by a TA before proceeding to implementation.

(1 pt) Write down and show a TA your trace through the solution for n_digits(721).

(2 pts) Implement your function. Update main() to read in a number from the user and call your function, then report the number of digits in their number. Try the test cases above plus new ones you make up.

Your function should have this format:

```
int n_digits(int n) {
    if ( . . . ) { /* base case 1 */
        return something;
    } else { /* recursive call */
        return something that uses n_digits(n/10);
    }
}
```

(1 pt) If you enter a number that is too big to fit into an integer, what happens? At what point does this happen (how big is n)? How could you change the function to work with larger numbers?

Discuss in your group: Would you find it easier or harder to count digits in an iterative fashion (e.g., with a loop)? What would that loop look like?

Get your work checked off by a TA.

Submit your program on Canvas (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 file from the ENGR servers to your local laptop.
2. Go to the Lab 9 assignment: https://canvas.oregonstate.edu/courses/1770357/assignments/7847931
3. Click "Submit Assignment".
4. Upload your .cpp file (lab9_recur.cpp).
5. Click the **"Submit Assignment"** button.
6. You are done!

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

Remember, the assignment you submit must be your work alone (no partners).
