CX 4010 / CSE 6010
Assignment 1
Combinatorics
Additional requirements for graduate students only
minor update to clarify leading 0's, 8/26/22

Initial Submission Due Date: 11:59pm on Thursday, September 1
Final Submission Due Date: 11:59pm on Thursday, September 8
Submit codes as a single zipfile as described herein to Canvas
48-hour grace period applies to all deadlines

In combinatorics, we often consider questions of how many ways there are to arrange items or how many items satisfy some criterion. In this assignment, you will consider a simple combinatorics question: how many nonnegative integers in a given range include a particular digit a specified number of times.

As an example, we may wish to count the number of nonnegative integers with up to three digits that include exactly two 7's. Combinatorics teaches ways of thinking to arrive at an answer: three-digit integers with exactly two 7's may take the form 77_ (9 options to fill in the blank without repeating a 7), 7_7 (9 options to fill in the blank), and _77 (9 ways to fill in the blank), for a total of 27 different possibilities. It can be a little more complicated if the range of integers to consider is narrower (e.g., if the goal is to count the number of integers from 123 to 765 that include exactly two 7's) and/or if more repetitions are specified.

Because the purpose of this assignment is to give you some experience programming in C, you will do this counting using "brute force" by examining every integer in the specified range to see if it meets the specified conditions.

Your assignment

Write a C program to count how many non-negative integers in a specified range from x to y (inclusive) satisfy the criterion of including exactly n m's, where m is a specified digit of interest from 0 to 9 (inclusive) and n is the number of repetitions of that digit.

Your program should read in these values in order: minimum of considered range, maximum of considered range, digit of interest, and number of repetitions. You can read in each value using a pair of statements like the following. Don't forget the `&` before the variable name in `scanf`; we will talk more soon about why this is necessary. Depending on your editor, you may need to erase and re-type the quotes if you choose to copy and paste. Note that you will need to declare the variable first, and it is not necessarily a good idea to name your variables something like `x`; this is just an example.

```
printf("Enter minimum of range: ");

scanf("%d",&x);
```

Your program should find the answer and display it on the screen along with an echo of the values that define the range, the digit of interest, and the number of repetitions of that digit of interest. You can use a statement like the following to display integer variables `x, y, m, n` and `count`

(again, you may not wish to use these same names for variables in your program). Depending on your editor, you may need to erase and re-type the quotes if you choose to copy and paste.

```
printf("The number of integers from %d to %d that include exactly %d %ds is %d.\n", x, y, n, m, count);
```

Although there are several ways to do this calculation, the purpose of this assignment is for you to get used to common C programming syntax and functionality. Thus, to accomplish the counting goal of this assignment, you should do the following.

- Include a loop over the specified range and consider each integer in that range sequentially. You may think of smaller integers as having leading zeros (e.g., 4 == 04 == 004 = …). However, to avoid any confusion, you may assume that the digit of interest specified (m above) will not be 0.
- When considering a particular integer, use integer arithmetic to extract each digit separately.
- Undergraduate students: assume that y, the maximum integer that will be specified for the range, is no more than 999,999 (six digits).
- Graduate students: assume that y, the maximum integer that will be specified for the range, may have any number of digits. Your code should determine the correct number of digits to consider based on the input value. To do so, you may need to use a mathematical function and thus may need to include the header file math.h, which includes definitions of many common mathematical functions, and to include the compiler instruction "-lm" at the end of your compile instruction line, which will instruct the linker to search the math library.

Your code must be well structured and documented to receive full credit. Be sure to include comments that explain your code statements and structure.

Final submission: You should submit to Canvas a single zipfile that is named according to your Georgia Tech login—the part that precedes @gatech.edu in your GT email address. For example, I would name my zipfile echerry30.zip.

The zipfile should include the following files:

(1) your code, named counting.c.

(2) a README text file (not formatted in a word processor, for example) that includes the compiler and operating system you used for compiling and running your code along with instructions on how to compile and run your program. (For this assignment, it is expected that this file will be quite short.)

(3) a series of three slides composed in PowerPoint or similar software, saved either in PowerPoint or as a PDF and named slides.pptx or slides.pdf, structured as follows:

- Slide 1: your name and a brief explanation of how you developed/structured your program. This should not be a recitation of material included in this assignment document but should focus on the main structural and functional elements of your program (e.g., the purpose of any loops you used, the purpose of any if statements you used to change the flow of the program, the purpose of any functions you created, etc.). In other words, assuming that the

mathematics behind what you are doing is already known, what were the main things you did to translate those requirements into code? You are limited to one slide.
- Slide 2: a brief statement explaining why you believe your program works correctly. For example, you could select a case or cases where you know the answer (explain how) and verify that your code achieves the correct answer. You are again limited to one slide.
- Slide 3: a brief exposition of what you felt were the most challenging one or two aspects of this assignment and how your final submission improves upon your initial submission, as appropriate—also limited to one slide.

Initial submission: Your initial submission does not need to include the slides. It will not be graded for correctness or even tested but rather will be graded based on the appearance of a good-faith effort to complete the majority of the assignment. No feedback will be given as you will have other opportunities for feedback during class or office hours as well as an additional week to finalize the assignment.

Some hints:

- Start early, and start small! Make sure you can compile and run something like a helloworld program first and make small, progressive modifications.
- Test the procedure for small ranges initially where you can identify the specific integers that satisfy the criteria.
- If reading the values in is uncomfortable for you, start by defining the values within the program, either directly using assign statements or using `#define`.
- Consider printing the specific integers that satisfy the criteria to the screen when you are testing/debugging. (But be sure you can turn this functionality off; large ranges could have hundreds or thousands of values to include in the count, and you would not want all of these to print to the screen!)
- If you have some combinatorics experience, you may wish to create some test cases from examples you have seen in that context.