Assignment #4: **Due date: 9 December, 2016**
Fundamentals of Programming **Total points 20**

Natural Prestidigitation

The natural world is full of hidden and beautiful mathematics. The whorls of a conch shell hide the Fibonacci sequence and its Golden Ratio, plants grow in fractal patterns, and comets trace hyperbolic patterns through the solar system. All those beautiful patterns hide in the grungy data of human observation.

In this assignment, you will write a program that determines the distribution of initial digits in a set of data. In the end, we want a program that reads in a list of positive numbers and outputs a list of 10 values: the frequency with which each digit 0–9 appears as the nth digit of one of the input numbers.

Task #1: Complete the functions

We'll break that problem down into easier steps. To do this task, open NaturalPrestidigitationFunctions.c in your editor, complete each step, or function mentioned below ONE by ONE. Execute your project to verify the completed function's working—it should PASS its corresponding test.

1. Write a function countDigits. countDigits(num) calculates the number of digits in the integer num. countDigits should evaluate to 1 for 0–9, 2 for 10–99, 3 for 100–999, etc. Hint: use repeated division by 10 to calculate the number of digits in num.

2. Write a function nthDigitBack. nthDigitBack(n,num) finds the nth lowest order digit in num, i.e., the nth digit from the right. We take the rightmost digit to be the 0th digit. nthDigitBack should evaluate to 0 for digits beyond the "start" of the number. Hint: use repeated division by 10 again, followed by the modulo operator to pick out just the rightmost digit. For example:

```
nthDigitBack(0,123) ⇒ 3
nthDigitBack(1,123) ⇒ 2
nthDigitBack(2,123) ⇒ 1
nthDigitBack(3,123) ⇒ 0
nthDigitBack(0,0) ⇒ 0
nthDigitBack(3,18023) ⇒ 8
```

3. Write a function nthDigit, using nthDigitBack and countDigits. nthDigit(n,num) finds the nth highest order digit of num, i.e., the nth digit from the left. We take the leftmost digit to be the 0th. nthDigit should evaluate to 0 for digits beyond the "end" of the number. For example:

```
nthDigit(0,123) ⇒ 1
nthDigit(1,123) ⇒ 2
nthDigit(2,123) ⇒ 3
nthDigit(3,123) ⇒ 0
nthDigit(0,0) ⇒ 0
nthDigit(3,18023) ⇒ 2
```

4. Write a function nthDigitTallyOne, using nthDigit. nthDigitTallyOne(n, num, tally) assumes that tally is a 10 element list tallying the number of nth digits seen so far. It updates tally to reflect the nth digit of num. In other words, if d is the nth digit of num, then increment the dth element of tally.

Created by. Steve Wolfman (University of British Columbia).

Examples:

```
nthDigitTallyOne(2, 1072, [0,0,1,2,0,0,3,0,9,0]) ⇒ [0,0,1,2,0,0,3,1,9,0]
nthDigitTallyOne(0, 2541, [0,0,1,2,0,0,3,0,9,0]) ⇒ [0,0,2,2,0,0,3,0,9,0]
```

5. Write a function nthDigitTally, using nthDigitTallyOne. nthDigitTally(n, nums) returns a tally of frequencies of 0–9 as the nth digits of all the numbers in nums.

   Here's a sample test case. These are enrollments in Research Triangle Park colleges and universities in Fall 2000.

Institution	Enrollment
Duke University	12176
North Carolina Central University	5476
Louisburg College (Junior College)	543
Campbell University	3490
University of North Carolina at Chapel Hill	24892
North Carolina State University	28619
Meredith College	2595
Peace College	603
Shaw University	2527
St. Augustine's College	1465
Southeastern Baptist Theological Seminary	1858

   Assume the variable enrollments contains the enrollment numbers from that table. Then:

```
nthDigitTally(0, enrollments) ⇒ [0,3,4,1,0,2,1,0,0,0]
```

When done writing all the above utility functions, you should see these messages in your console window.

```
Running tests ...

PASSED. CountDigits()
PASSED. nthDigitBack()
PASSED. nthDigit()
PASSED. nthDigitTallyOne()
PASSED. nthDigitTally()

All tests complete.
```

Task #2: Find Hidden Pattern in a Data Source!

Comment the runTests() line in NaturalPrestidigitation.c, and uncomment the call to readMysteriousNumbers() function. The functions reads in integers from an input comma-separated file (terminated by end-of-file) and returns a list of the numbers suitable as input to nthDigitTally(). The data is a selected chunk of education indicators compiled for Pakistan from Pakistan Data Portal website.

Using this data, readMysteriousNumbers() should produce the output similar to,

```
 *** Mystery numbers ***

0s: 5136 (55%)
1s: 499 (5%)
2s: 446 (4%)
3s: 457 (4%)
4s: 446 (4%)
```

```
5s: 482 (5%)
6s: 469 (5%)
7s: 462 (4%)
8s: 441 (4%)
9s: 439 (4%)
```

Optional: Find a Data Source to Use!

If you want to find the patterns hidden in the numbers around you, try the following: Find a data source on the web that no one else has used and transform it into a format suitable for input to readMysteriousNumbers. The data must all be separate measurements of a single type of phenomenon. For example: measurements of university/college enrollments across different institutions or at the same institution across different years; measurements of the flow rates of major rivers; measurements of the height of 10000 randomly chosen residents; measurements of the number of hits per day on a website over three years; measurements of the length in characters of each article in the Wikipedia; measurements of the population of the 1000 largest cities; etc. Furthermore, there must be at least 250 measurements in the list (but more would be better!).

Hand-in

This is a TEAM project. Form one with at most 2 (two) people (team with single member is also allowed). Designate one member as the "team leader", who will be responsible for the submission of the project.

Hand-in ONLY the files listed below at the appropriate location on the blackboard system at LMS. You should hand in a single compressed/archived file named Assignment_4_<your reg. No. XXX without angle brackets>.zip that contains the following files.

1. COMMENTED source file (NaturalPrestidigitationFunctions.c) with all the completed utility functions.
2. CLEAR snapshot of your console (SNAPSHOT.png) to show that your code is working.
3. PLAIN text file (OUTPUT.txt) that includes a) author information at the beginning, b) a brief explanation of the assignment, c) the number pattern found, d) the law outlining the pattern, e) data source used for the optional task and the result obtained, and e) any comments, or suggestions.

Programs will be graded on the following criteria:

1. **Program Specifications / Correctness.** does it compile? Are there obvious errors? Are there subtle errors? (25%)
2. **Documentation.** Is your program consistently indented in a manner that reflects the structure of your code? Is it easy to read? Are there blank lines which break up the major sections of your code? (15%)
3. **Efficiency.** Is your program efficiently organized, or is there a lot of duplicated code? Does it look well-written, or barely finished? (5%)
4. **Assignment Specifications.** Does your program fulfill the basic requirements? Is it done? And what else does it do? (15%)

5. **In time.** (10%)
6. **Plagiarism.** (30%)

Honor code

The student should agree to the terms below; any infringements will result in minimum marks.

- will not cheat on project
- will not share solutions to the project; and
- will notify the instructor immediately if he or she becomes aware of any other group cheating