EE 312

Assignment 1

Code Camp

Due: Friday, September 4 at 11:59pm

100 points

The purpose of this assignment is for you to:

1. Learn an IDE, write a program, run a program (in Windows/Mac and Linux), and turn in the program to Canvas.

2. Review the fundamentals of C.

3. Write functions with parameters and return values

4. Practice using one-dimensional arrays

5. Experience writing test cases and using test scripts

6. Practice with pre- and post-condition

7. Use makefiles

Most of all, the purpose of this project is for you to become familiar with the software and some of the tools you will need in EE 312, as well as the expectations for projects for this course.

Provided File(s): CodeCamp.zip contains the following:

- CodeCamp.c (function shells)
- CodeCampTester.c (main function used for testing)
- makefile

Assignment Restrictions:

- Your code must be written in C (e.g., not C++).
- Your code must work when compiled with gcc and run on the linux machine kamek.ece.utexas.edu.

Getting started:

1. Go to https://utdirect.utexas.edu/apps/ece/db/acme to enable your ECE Linux account. You will need this account to access the department's Linux machines.

2. If you have not done so, sign up for the class discussion group on Piazza. You will need to check messages on Piazza regularly for announcements and Q&A.

3. Install an IDE that will compile C on your machine. We will be using CLion and gcc in class.

4. Remember that we will grade your program on kamek.ece.utexas.edu using gcc on the command line. If your program does not compile and run in this environment, we can't grade it. Test your program before you submit it.

5. Download CodeCamp.zip and extract the files.

6. Complete the functions in CodeCamp.c. See the function descriptions below.

7. Add at least two tests per function except sum3or5Multiples. You may copy the structure of the provided tests, but change the data and expected results. You must write your own test cases to turn in, but you can share test cases on Piazza to help others test their program. Before you turn in CodeCampTester.c, delete the original tests, except the one for sum3or5Multiples.

8. Fill in the header for CodeCamp.c. Replace <NAME> with your name. You are stating, on your honor, that you did the assignment on your own, as required and did not share your code with anyone else. If you copy code from someone else or give your code to someone else you will be subject to academic dishonesty penalties as described on the course syllabus. Fill in the header at the top of CodeCampTester.c.

9. Create a zip file named prog1_<YOUR_EID>.zip (case sensitive - do not name the file Prog1.zip) with your CodeCamp.c and CodeCampTester.c files. The zip file must not contain any directory structure - just the two required files. With zip, use the -j option.

10. Turn in prog1_<YOUR_EID>.zip via your Canvas account to programming project 1.

11. Ensure your files are named CodeCamp.c and CodeCampTester.c. Failure to do so will result in significant point deductions.

12. Make sure that you follow the style guide for EE 312. You will lose points if you do not.

Getting help: Start the project early in order to get help. You must complete this assignment on your own. You may receive help during office hours and on the class discussion group on Piazza (for general questions only).

Checklist: Did you remember to:

1. Review and follow the general assignment requirements?

2. Work on the assignment by yourself and complete all the solution code on your own?

3. Fill in the headers in the files CodeCamp.c and CodeCampTester.c?

4. Implement the four required functions?

5. Ensure that your program compiles and executes correctly on the department 64-bit linux machines?

6. Make sure that your code passes the tests provided in CodeCampTester.c? Note: passing all the provided tests does not mean your solution code is correct. We will use other unpublished tests when we grade your program.

7. Remove the tests that were provided to you in CodeCampTester.c, except for the test for the sum3or5Multiples function?

8. Turn in your files in a zip named prog1_<YOUR_EID>.zip with no internal directory structure?

9. Turn in your zip named prog1_<YOUR_EID>.zip to Programming Project 1 on Canvas?

Notes:

You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside the class. Request help from the instructor or TAs.

Make sure you follow the style guide for EE 312 projects.

Function Descriptions:

1. Hamming Distance: "The Hamming distance between two strings of equal length is the number of positions for which the corresponding symbols are different. Put another way, it measures the number of substitutions required to change one into the other, or the number of errors that transformed one string into the other." From the Wikipedia article on Hamming Distance. For this problem you will be working with arrays of ints instead of strings.

```
/* Determine the Hamming distance between two arrays of ints.

    pre: aList != NULL, bList != NULL, aList contains len
elements bList contains len elements, len > 0
```

```
    post: return the Hamming distance between the two arrays of
ints.

    Neither the parameter aList or bList are altered as a result
of this function.

*/

int hammingDistance(int aList[], int bList[], int len) {
```

Example: For array {1, 2, 3, 4} and array {3, 2, 3, 7}, the Hamming distance is 2.

2. lastDigit: takes two ints and returns true if they have the same last digit, and false if they do not have the same last digit.

```
/* Determine if two ints have the same last digit.

   Note that the last digit of 0 is 0 and the last digit of -12
is 2.


   post: return true if x and y have the same last digit, false
otherwise.

*/

bool lastDigit(int x, int y) {
```

Example: lastDigit(0, 120) returns true (1). lastDigit(-24, 1534) returns true. lastDigit(12, 3115) returns false.

3. sum3or5Multiples: computes and returns the sum of all the positive integers less than 1000 that are a multiple of 3 or 5.

```
/* Determine the sum of the positive integers less than 1000
that are multiples of 3 or 5 (or both).

    post: Return the sum of the integers from 1 to 1000 that are
multiples of 3 or 5 or both.

*/
```

```
int sum3or5Multiples() {
```

4. reverseInt: takes an int x and returns the value with the digits reversed, if the reversed value can be stored as an int.

```
/* Reverse the digits of an integer. Return the reversed number
if it is in the range of type int, and 0 otherwise.

    post: return x with digits reversed if the reverse can be
stored as an int, 0 otherwise.

*/

int reverseInt(int x) {
```

Example: reverseInt(-143) is -341. reverseInt(200) is 2. reverseInt(30001) is 10003.

Attribution: Roger Priebe