Lab 5 - Functions

Due: end of class

OVERVIEW

For this project, you will practice decomposing a problem into functions. In the final part, you will also get some practice with pass-by-reference functions.

As part of this lab is to also gain experience with integer operations, you are NOT allowed to convert numbers into strings and use string processing primitives.

TASK 0

Consider the standard decimal representation of a non-negative integer num. You wish to write a function that repetitively adds up the digits of a number until it eventually results in a single-digit number. For example, given the number 234567, the first iteration results in 27 (=2+3+4+5+6+7), and the 2nd iteration results in 9 (=2+7). Since 9 is a single-digit number, no more iterations are needed and 9 is returned.

Write a function with the following prototype to do this:

```
// Precondition: num > 0
// Postcondition: the return value is the iterated sum of digits of num
int sumDigits(int num);
```

To do this, we first need to identify functions that are useful to doing this. In this case, a list of useful function prototypes might be:

- `int getDigit(int num, int index);  // return the index'th digit of num`
- `int numDigits(int num);            // return the number of digits in num`

You should have at least these three functions, though you may have additional functions if you wish. Don't forget to write pre/post conditions for these functions.

You may use library functions defined in cmath (such as pow) if you wish, though you don't need to.

TASK 1

You need a driver program to test your above program. To do this, write a main function that prompts for and input num, calls sumdigits and outputs the result. The program should also do error checking to ensure that inputs are legal, and reprompt as needed.

TASK 2

A well-known theorem of math states that the above sum is 9 iff the number is divisible by 9. Write a function that modifies its argument number so that its divisible by 9. The function should add something to the right-most digit of the number if possible; otherwise it should subtract something from that digit. An example use of this function might be:

```
transformNum(n);
cout << n;  // prints 234567 if n was originally 234565
```

Write the transformNum function. You should be able to reuse most of the code from earlier. Also modify the driver program to output the transformed number.

HAND IN

Your 136 instructor will tell you what to hand in and how.

GENERAL COMMENTS FOR ALL PROGRAMS THIS SEMESTER

You should have the following header on all programs:

```
/*
  Author: <name>
  Course: {135,136}
  Instructor: <name>
  Assignment: <title, e.g., "Lab 1">

  This program does ...
*/
```

GRADING

All 135 and 136 programs this semester will be graded on:

- Correctness: Does your program work?
- Testing: Have you generated sufficient and good test data to give reasonable confidence that your program works?
- Structure: Have you structured your code to follow proper software engineering guidelines? This includes readability and maintainability.
- Documentation: How well documented is your code? Good documentation does not repeat the code in English, but explains the point of each code block, highlighting any design decisions and/or tricky implementation details.