McMaster University

COMPENG 2SH4
Principles of Programming
LAB 1
Instructor: Mohamed Hassan
Fall 2022

This lab is worth 7% of the course mark.
The number in square brackets at the beginning of each question shows the number of points the question is worth. The total number of points is 70.

General Requirements on Your Programs

A. You will be using the exact same process from Lab0 to accept the invitation➔clone the git repo➔create the eclipse project➔develop your lab code➔push to the repo.
B. All lab questions have to be implemented in the provided Questions.c template.
C. You have to follow the guidelines in the provided comments such that your code passes the tests.
D. Your programs should be written in a good programming style, including instructive comments and well-formatted, well-indented code. Use self-explanatory names for variables as much as possible. (~5% of the mark)
E. You have to make sure you pass all the tests. Please note that passing the tests does not grant you automatically the full mark, we run other "hidden" test cases that are not shared with you to further assess your code. You are required to make sure your work is correctly implemented to the best of your knowledge. One task that can help you with that is to add further tests to stress the corner cases of each question.
F. For each question, you are required to add **at least** one additional test to the testCases.c file.

Lab Github Classroom Invitation Link

https://classroom.github.com/a/mTU5DtH5

Lab Deadline

The deadline for lab1 is Sept 30th . Please note that this is a programmed deadline in the environment, so make sure you submit in time since you will not be able to submit after that dictated deadline.

Lab Questions

For this lab and all other labs, please make sure you do not use *printf(), nor scanf().*

1. [6] complete the following functions in the Questions.c file:

   - `int Q1_for()`
   - `int Q1_while()`
   - `int Q1_do()`

   to compute the sum of all numbers that are multiples of 4, between 30 and 1000 (including the limits), in three different ways: with a for loop, a while loop and a do-while loop, accordingly. Return the total sum at the end of each function.

2. [7] For this exercise you should be able to write a logical expression (i.e., with logical operators) which checks if some integer x consists of exactly 5 digits. Ex: 30498 and -14004 are 5-digit numbers, while 1098, -1 and 34 are not.

   Complete the `int Q2(int Q2_input)` function that takes an input integer parameter and returns `1 if the number is five digits and 0.`

3. [7] Complete the function `int Q3(float Q3_input)` that takes a student's average as an input, which is a floating-point value, and returns:

   - 4 if the average is in the range 90-100,
   - 3 if it is in the range 80-89,
   - 2 if it is in the range 70-79,
   - 1 if it is in the range 60-69 and
   - 0 if the average is between 0 and 59.

   If the average is not in the range 0-100, the program should return -1.

4. [10] Calculate the value of π from the infinite series

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \ldots$$

   Complete the `double Q4(int Q4_input)` function which reads a positive integer `Q4_input` as an input parameter and calculates the value of π by adding up the first `Q4_input` terms of the above series. You have to return the calculated value as a double.

5. [10] (Pythagorean Triples) A right triangle can have sides that are all integers. The set of three integer values for the sides of a right triangle is called a Pythagorean triple. These three sides must satisfy the relationship that the sum of the squares of two of the sides is equal to the square of the hypotenuse.

   Complete function `int Q5()` to find all Pythagorean triples for side1, side2 and the hypotenuse all no larger than 400, with side1<= side2. Use a triple-nested for loop that simply tries all possibilities. This

is an example of the "brute-force" approach. Finally, you must return the total number of triples at the end.

6. [15] A positive integer number is said to be a ***perfect number*** if its positive factors, including 1 (but not the number itself), sum to the number. For example, 6 is a perfect number because 6=1+2+3.

Complete the `int Q6(int Q6_input, int perfect[])` function that determines all perfect numbers smaller than or equal to some integer `Q6_input` (`Q6_input` > 1).

- The array `perfect[]` should hold all the perfect numbers you find.
- The function should also return the total count of the perfect numbers you found.

Note: Assume that x and y are two positive integers. Then x is a **factor** of y if the remainder of the division of y by x is 0. For instance, 5 is a factor of 15, but not of 36.

For example: if `Q6_input` is 10 then the only perfect number you will find is 6. Accordingly, `perfect[0]` should be equal 6 and the function should return 1 as your count.

7. [15] **a)** Complete the `int Q7a(int Q7_input)` function takes a seven-digit positive integer as input and returns it reversed. For example, if the integer is 9806593, the function should return 3956089.

   You are not allowed to use any function of C standard library. You are not allowed to use arrays either. For the case when the integer ends with 0, the number returned cannot have leading 0's (Eg: input 3412400; output 42143).

   Hint: Use the division and remainder operators to separate the number into its individual digits.

   b) Modify your program so it returns backwards any positive integer, not necessarily a seven-digit one.