Homework: Different Digits

# Overview

In a distant country, a queen is very superstitious and only likes numbers with repeated digits like 77, 434, or 2121. Thus, she declares any number without repeated digits to be invalid. You have been appointed by Her Majesty to write a program that, given two integers ***a*** and ***b,*** determines how many valid numbers exist in the range between ***a*** and ***b,*** inclusive.

Examples:

* “Repeated” here means “appears more than once”
  + 77 has repeated 7s.
  + 434 has repeated 4s.
  + 2121 has repeated 1s and 2s.
* For **a = 3** and **b = 12**, the answer is **1**.
  + The numbers that exist between **a** and **b** are: 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
  + Of those ten numbers, one is valid (has repeated digits): 11
* For **a = 12** and **b = 34*,*** the answer is **2**.
  + The numbers that exist between 12 and 34 are: 12, 13, 14, 15, 16, 17, 18, 19, 29, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34.
  + Of those twenty-three numbers, two are valid (have repeated digits): 22 and 33.
* For **a = 1** and **b = 100,** the answer is…
  + Let’s see…
  + The numbers between 1 and 100 are: 1,2,3,4,5,6,7,8,9,10,11,...,22,..,33,...,44,...,55,...,66,...,77,...,88,...,97,98,99,100
  + Of those one hundred numbers, ten are valid (have repeated digits): 11, 22, 33, 44, 55, 66, 77, 88, 99, and 100
  + So, the answer is **10**.

# Objectives

1. Practice the use of standard input and output;
2. Practice using selections, iterations, and simple functions;
3. Observe the importance of designing test cases that cover the space of valid input values.
4. Separate code into multiple files.
5. Practice solving problems.

# Requirements

When developing your solution to this problem, ensure that your program conforms to the following requirements and assumptions:

1. Name the source file containing the main function different\_digits.cpp. Name the source file containing the function declarations functions.h. Name the source file containing the function definitions functions.cpp. These files already exist in the starter code.
2. Implementation is written such that it is readable by other programmers. Use descriptive variable identifiers and comments where appropriate (comments should explain things that are not obvious from the code).
3. You cannot use the string class in this assignment. Neither can you use data structures not yet covered in this course such as vector. You could use an array, but you shouldn’t.[[1]](#footnote-0)
4. The input/output format should be exactly as follows. The expected input to the program is given by two integers a and b such that 0 < a <= b <= 10000.

Required i/o format (user input in **bold red**; everything else is output):  
   
 Enter numbers 0 < a <= b <= 10000: **3 12**↵

There are 1 valid numbers between 3 and 12↵

Enter numbers 0 < a <= b <= 10000: **12 34**↵

There are 2 valid numbers between 12 and 34↵

Enter numbers 0 < a <= b <= 10000: **1 100**↵

There are 10 valid numbers between 1 and 100↵

Enter numbers 0 < a <= b <= 10000: **-7 10**↵

Invalid input↵

Enter numbers 0 < a <= b <= 10000: **12 100230**↵

Invalid input↵

Enter numbers 0 < a <= b <= 10000: **1 5**↵

There are 0 valid numbers between 1 and 5↵

Enter numbers 0 < a <= b <= 10000: **121 122**↵

There are 2 valid numbers between 121 and 122↵

1. Your code should check if the input is valid: the numbers are positive integers, the second one is greater than or equal to the first, and so on. If the input does not meet the requirements, the output should be “Invalid input”, followed by a re-prompt for input. The program should terminate after providing the result for valid input (see the test cases on Mimir).
2. Your program must define and use the following functions:

**bool is\_valid\_range(int a, int b):** This function returns the boolean value true if and only if inputs a and b satisfy the constraint that 0 < a <= b <= 10000.

**is\_valid\_range(12, 34)** should return true.

**is\_valid\_range(34, 12)** should return false.

**int count\_digit\_occurrences(int number, int digit):** This function returns how many times digit occurs in number, where 0 <= digit <= 9. The value of number can be *any* integer.

**count\_digit\_occurrences(434, 3)** should return 1.

**count\_digit\_occurrences(434, 4)** should return 2.

**count\_digit\_occurrences(434, 5)** should return 0.

**count\_digit\_occurrences(0, 3)** should return 0.

**count\_digit\_occurrences(-434, 4)** should return 2.

**bool has\_repeated\_digits(int number):** This function returns the boolean value true if and only if number contains any digit which appears more than once.

**has\_repeated\_digits(7127)** should return true.

**has\_repeated\_digits(7128)** should return false.

**int count\_valid\_numbers(int a, int b):** This function returns how many numbers in the range [a, b] (i.e. a <= number <= b) are valid according to the queen’s superstition, i.e. how many numbers contain repeated digit).

**count\_valid\_numbers(12, 34)** should return 2. (22 and 33)

**count\_valid\_numbers(1, 5)** should return 0.

These functions must be defined (and, therefore, submitted) in a separate file functions.cpp along with the corresponding header file functions.h which contains the declarations of these functions. This header file must be included at the beginning of different\_digits.cpp and functions.cpp. For example, different\_digits.cpp should begins with:

#include <iostream>

#include "functions.h"

This is already done for you in the starter code.

A convenient way to compile multiple source files (in this case the two files: functions.cpp and different\_digits.cpp) is to put all the source and header files in a directory (for example, hw\_different\_digits) and run g++ on all source files in this directory:

1. Download the starter code from Mimir.
2. Make a directory named hw\_different\_digits.
3. Put the starter code in hw\_different\_digits.
4. In a terminal (assuming you are in the parent directory of hw\_different\_digits):

$ cd hw\_different\_digits

$ ls

different\_digits.cpp functions.cpp functions.h

$ g++ -std=c++17 -Wall -Wextra -pedantic -Weffc++ \*.cpp

The \* symbol is a wildcard that matches any valid character in an identifier. So, \*.cpp means all files in the current directory whose name ends in .cpp (i.e. all C++ source files).

1. Your program must compile without errors or warnings.

# Getting Started

1. Download the starter code.
2. Compile and run it.
3. Submit it to Mimir.
4. Implement is\_valid\_range()
5. Recompile and rerun.
6. Resubmit to Mimir.
7. Continue writing just enough code to pass the next test on Mimir (in order 1 -- 20)

# Submission

Submit these three files to Mimir:

* different\_digits.cpp
* functions.cpp
* functions.h

1. It is important to get fluency in solving problems in ways that lead to efficient usage of memory. The thinking process will be useful in solving larger problems when you do use the new data structures we cover later in this course. [↑](#footnote-ref-0)