This is an individual assignment.  Do not share your code with other students.  Do not show your code to other students.  Do not look at the code of other students.  Do not ask other students how they solved a problem.  HWs are **individual** practice assignments (LWs, on the other hand, are collaborative practice assignments).  If you have questions about this assignment, talk to a Peer Teacher, a TA, or an instructor.

* [Go to Mimir to submit your code (Links to an external site.)](https://class.mimir.io/cas_login/5315e0bf-a5bc-4521-baec-923474543dd5)
* [Ask questions on Campuswire (Links to an external site.)](https://campuswire.com/c/G4A9C0412/feed).
  + Look here to see if someone has already asked your question.
  + It is very rare that a student's question is unique. Ask here and everyone can benefit from the answer.
  + If you know the answer, go ahead and answer it yourself!
  + Do not post HW code publicly.  Make a private post to Instructors and TAs.
    - Do not post screenshots of code.  Make a code block in your post and paste your code into it.

If you know about another student who is sharing their code with other students (or in any other way is violating the Aggie Code of Honor), you should report them to the instructor or [report them to the AHSO (Links to an external site.)](https://cm.maxient.com/reportingform.php?TexasAMUniv&layout_id=11) (https://cm.maxient.com/reportingform.php?TexasAMUniv&layout\_id=11).

**Overview**

**Dr. Eva Thomas is an underwater geographer studying mountains and valleys in the Atlantic Ocean. She models mountain and valley ranges using integers whose digits alternate between increasing and decreasing. You have been hired to write a program that, given two integers *a*and *b,*determines how many numbers model mountain and valley ranges in the range between *a* and *b,* inclusive.**

**Examples:**

* + An *n*-digit integer, for *n*>=2, with digits d1d2d3d4… models a mountain range if it has the pattern /\/..., that is , d1 < d2, d2 > d3, d3 < d4 etc., and it models a valley range if it has the pattern \/\..., that is d1 > d2, d2 < d3, d3 > d4, etc.
    - 1503 has 1 < 5, 5 > 0, and 0 < 3, so it models a mountain range.
    - 7120 has 7 > 1, 1 < 2, and 2 > 0, so it models a valley range.
    - 7113 has consecutive 1s, so it models neither.
  + For **a = 15**and **b = 25**, the answer is **8 mountain ranges**and **2 valley ranges**.
    - The numbers 15, 16, 17, 18, 19, 23, 24, and 25 are mountain ranges.
    - The numbers 20 and 21 are valley ranges.
    - The number 22 is neither a mountain nor a valley range.
  + For **a = 1234** and **b = 4321*,***the answer is **753 mountain ranges**and **351 valley ranges**.
  + For **a = 10** and **b = 1000,** the answer is **276 mountain ranges**and **330 valley ranges**.

See the following how we got the solution for a = 15, and b = 25, and how to handle 2, 3 or 4 digits:

**2 Digit Value**

Mountain (M): if d1 < d2

Valley (V): if d1 > d2

Neither (N)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Count** | **Value** | **d1** | **d2** | **Result** |
| 1 | 15 | 1 | 5 | M |
| 2 | 16 | 1 | 6 | M |
| 3 | 17 | 1 | 7 | M |
| 4 | 18 | 1 | 8 | M |
| 5 | 19 | 1 | 9 | M |
| 6 | 20 | 2 | 0 | V |
| 7 | 21 | 2 | 1 | V |
| 8 | 22 | 2 | 2 | N |
| 9 | 23 | 2 | 3 | M |
| 10 | 24 | 2 | 4 | M |
| 11 | 25 | 2 | 5 | M |

**3 Digit Value**

Mountain (M): if d1 < d2, d2 > d3

Valley (V): if d1 > d2, d2 < d3

Neither (N)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Count** | **Value** | **d1** | **d2** | **d3** | **Result** |
| 1 | 123 | 1 | 2 | 3 | N |
| 2 | 124 | 1 | 2 | 4 | N |
| 3 | 125 | 1 | 2 | 5 | N |
| 4 | 126 | 1 | 2 | 6 | N |
| 5 | 127 | 1 | 2 | 7 | N |
| 6 | 128 | 1 | 2 | 8 | N |
| 7 | 129 | 1 | 2 | 9 | N |
| 8 | 130 | 1 | 3 | 0 | M |
| 9 | 131 | 1 | 3 | 1 | M |
| 10 | 132 | 1 | 3 | 2 | M |
| 11 | 133 | 1 | 3 | 3 | N |
| 12 | 134 | 1 | 3 | 4 | N |

**4 Digit Value**

Mountain (M): if d1 < d2, d2 > d3, d3 < d4

Valley (V): if d1 > d2, d2 < d3, d3 > d4

Neither (N)

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Count** | **Value** | **d1** | **d2** | **d3** | **d4** | **Result** |
| 1 | 7623 | 7 | 6 | 2 | 3 | N |
| 2 | 7624 | 7 | 6 | 2 | 4 | N |
| 3 | 7625 | 7 | 6 | 2 | 5 | N |
| 4 | 7626 | 7 | 6 | 2 | 6 | N |
| 5 | 7627 | 7 | 6 | 2 | 7 | N |
| 6 | 7628 | 7 | 6 | 2 | 8 | N |
| 7 | 7629 | 7 | 6 | 2 | 9 | N |
| 8 | 7630 | 7 | 6 | 3 | 0 | N |
| 9 | 7631 | 7 | 6 | 3 | 1 | N |
| 10 | 7632 | 7 | 6 | 3 | 2 | N |
| 11 | 7633 | 7 | 6 | 3 | 3 | N |
| 12 | 7634 | 7 | 6 | 3 | 4 | N |

**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 mountains\_valleys.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 or functions from the <algorithms> library. You could use an array, but you shouldn’t.

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  10 <= a <= b < 10000.

Required i/o format (user input in **bold red**; everything else is output):

Enter numbers 10 <= a <= b < 10000: **15 25**↵

There are 8 mountain ranges and 2 valley ranges between 15 and 25.↵

Enter numbers 10 <= a <= b < 10000: **1234 4321**↵

There are 753 mountain ranges and 351 valley ranges between 1234 and 4321.↵

Enter numbers 10 <= a <= b < 10000: **10 1000**↵

There are 276 mountain ranges and 330 valley ranges between 10 and 1000.↵

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

Invalid Input↵

Enter numbers 10 <= a <= b < 10000: **12 100230↵**

Invalid Input↵

Enter numbers 10 <= a <= b < 10000: **110 115↵**

There are 0 mountain ranges and 0 valley ranges between 110 and 115.↵

5. 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).

6. 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 10 <= a <= b < 10000.

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

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

**char classify\_mv\_range\_type(int number):** This function returns the char value ‘M’ if number models a mountain range, ‘V’ if number models a valley range, and ‘N’ if number does not model either.  The input is **not** constrained by the range in Requirement 4.

**classify\_mv\_range\_type(1503)**should return ‘M’.

**classify\_mv\_range\_type(7120)**should return ‘V’.

**classify\_mv\_range\_type(7113)**should return ‘N’.

**void count\_valid\_mv\_numbers(int a, int b):** This function prints out how many numbers in the range [a, b] (i.e., a <= number <= b) are mountain ranges and valley ranges using the definition in the Overview. This function is **not** constrained by the range specified in Requirement 4.

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 mountains\_valleys.cpp and functions.cpp. For example, mountains\_valleys.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 mountains\_valleys.cpp) is to put all the source and header files in a directory (for example, hw\_mountains\_valleys) and run g++ on all source files in this directory:

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

$ cd hw\_mountains\_valleys

$ ls

mountains\_valleys.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).

7. Your program must compile without errors or warnings.

Getting Started

1. Download the starter code. The starter code is available on Mimir. To get the starter code go to Mimir and select the [HW] Mountains and Valleys. Click the “Download Starter Code” button.
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)

Number Peeling

You might like to know of this interesting technique called "number peeling".  Given a decimal (base-10) number like 8675309, we can read each digit, one-by-one as follows:

1. number <-- 8675309  
2. while number is not 0 do  
3. digit <-- remainder after dividing number by 10  
4. number <-- number divided by 10  
5. end

If you trace this algorithm, you will see that *digit*takes on the values 9, 0, 3, 5, 7, 6, 8, in that order (i.e. the digits in the number as read from right to left, what you might call reverse order).  You can peel numbers in any base by simply changing the divisor and modulus, e.g. to peel in base 7, take remainder after division by 7 and divide by 7 at each step.

Submission

Submit these three files to Mimir:

* mountains\_valleys.cpp
* functions.cpp
* functions.h