Problem Set 2 Exercise #24: Reverse Number

Reference: Lecture 5 notes

Learning objectives: Repetition statements; Algorithm design

Estimated completion time: 60 minutes

Problem statement:

[Past year CS1101 Sit-in Lab Question]

You are to write a program **reverse_number.c** that reads two positive integers *low* and *high*, and finds out how many integers in the range [*low*, *high*] (both inclusive) whose reverse is the same as the original value. You may assume that *low* ≤ *high*.

For example, if *low* is 150 and *high* is 202, then there are 6 integers in the range [150, 202] whose reverse is the same as itself. They are: 151, 161, 171, 181, 191, and 202.

You need to write a modular program. Besides the **`main()`** function, there should be at least another function that computes some result.

You are **not** allowed to use array or string functions for this task.

Useful tips:

The challenge is how to get the reverse of a given integer. Hardcode different cases (has 1 digit, 2 digits, 3 digits...) is not the proper way of problem solving.

Engage a loop to extract digits round by round, along the way accumulate the reverse number. When the loop ends, reverse number should be created. Of course, it's easier to say than to do it ☺.

Sample run #1:

```
Enter the range: 150 202
There are 6 reverse numbers.
```

Sample run #2:

```
Enter the range: 12 21
There are 0 reverse numbers.
```