Exercise 3:

(1) The sum of an array is the sum of its individual elements. For example, if an array is numbers = {1, 2, 3, 4}, the sum of the array is 1+2+3+4 = 10.
Function Description: Complete the function summation. The function must return the integer sum of the numbers array.

int summation(int numbers_size, int* numbers)

(2) Write a function which, given a string, return TRUE if all characters are distinct and FALSE if any character is repeated.

(3) Write a C function that use the bubble sort algorithm to sort an integer array in ascending order (search for the bubble sorting algorithm).

(4) Write a C function that use the selection sort algorithm to sort an integer array in ascending order (search for the selection sorting algorithm).

(5) Write a C function to return the index of **FIRST** occurrence of a number in a given array. Array index start from 0. If the item is not in the list return -1. (Linear Search Algorithm)

Example:
Array = {1,2,3,4,4,4}
The required number is 4 it should return 3

(6) Write a C function to return the index of **LAST** occurrence of a number in a given array. Array index start from 0. If the item is not in the list return -1. (Linear Search Algorithm)
Example:

Array = {1,2,3,4,4,4}
The required number is 4 it should return 5

(7) Write a program that computes the nth term of the arithmetic series:
1, 3, 5, 7, 9, …
Run the program to compute the 100th term of the given series.

(8) Write a program that computes the nth term of the geometric series:
1, 3, 9, 27, …
Run the program to compute the 10th term of the given series.

(9) The sequence of numbers 1, 1, 2, 3, 5, 8, 13, … is called Fibonacci numbers; each is the sum of the preceding 2. Write a program which given n, returns the nth Fibonacci number.
- with for/while
- with recursion

(10) Write a function which, given a string, converts all uppercase letters to lowercase, leaving the others unchanged.

(11) Write a function that prints the frequency of a certain character in a string.

(12) Write a function to find the length of a string.

(13) Write a function to remove all characters in a string expect alphabet.

(14) Write a function to reverse a string by passing it to a function.

(15) Write a function to concatenate two strings without using strcat function.

(16) Write a C function that takes an array as input and reverse it.
Example:
Input : 1,2,3,4,5
Output: 5,4,3,2,1

(17) Write a C Program for swapping two arrays "A & B" with different lengths. B will be always the smallest array.

int * Swap (int a_size,int *a,int b_size,int *b)

(18) Write a C function that return the count of the longest consecutive occurrence of a given number in an array.
Example:
Array={1,2,2,3,3,3,3,4,4,4,4,3,3,3} and searching for 3 → result = 4

(19) Write a C function that compare between 2 arrays with the same length. It shall return 0 if the two arrays are identical and 1 if not.

(20) Write a C function to return an array containing the values between two 8-bits unsigned integers **IN DESCENDING ORDER EXCLUSIVE**. The function takes 2 values (**LowerValue** and **UpperValue**), it shall determine the values in between, and then arrange the sequence in descending order excluding the upper and lower values.
If the LowerValue is greater than or equal the UpperValue, return an array of 2 elements, both containing value = 0xFF

Example:
Input: LowerValue=2 and UpperValue=5
Output:
Output Array=4,3
Output Array Size=2

(21) Write a C function to return an array containing the values between two 8-bits unsigned integers **IN DESCENDING ORDER INCLUSIZE.** The function takes 2 values (**LowerValue** and **UpperValue**), it shall determine the values in between, and then arrange the sequence in descending order including the upper and lower values.
If the LowerValue is greater than the UpperValue, return an array of 2 elements, both containing value = 0xFF

Example:
Input: LowerValue=2 and UpperValue=5
Output:
Output Array=5,4,3,2
Output Array Size=4

(22) Write a c function that removes the repeated number of an input sorted array and return a new array without the repeated numbers. The function shall return error if the size of the input array is ZERO. The function takes four inputs:

a. Old array.
b. Old array size.
c. New array (empty array).
d. The size of the new array after fill it in the function.

int removeDuplicates(int arr_old[], int n_old, int arr_new[], int *n_new)

Example:

arr1 = {1,2,3,3,3,4,4,5,5,5} → arr2 = {1,2,3,4,5}