Faculty of Arts & Sciences
Department of Computer Science
CMPS 270 – Software Construction
Fall 2022 – Assignment 1

Guidelines and Grading Policy

A program that does not compile will receive a **zero**. For every exercise, make sure you upload your solutions to your GitHub repository and maintain proper version control practices (commit your work regularly and meaningfully). You are also required to include the specifications of each method, in addition to the testing strategy and test-cases used to validate your program's correctness. The GitHub, specifications, and testing skills will account for 50% of the grade. The remaining 50% is for the program's correctness.

Exercise 1. Hamming Distance **(call your program hamming_distance.c)**

Create a C program that asks the user to enter pairs of positive integers (separated by a space), and a negative (non-zero) integer to stop. For each pair of integers, it will print the Hamming distance between the bit representations of those two integers. For example, if the user inputs 0 1, then the Hamming distance is equal to 1. The Hamming distance between two strings is the number of positions in which they differ. The bit representation of 0 is ....0000 and the bit representation of 1 is ....0001 (where dots represent all zero). So the Hamming distance between them is 1. The Hamming distance between 0 and 3 would be 2. The Hamming distance between 1 and 3 would be 1.

```
Example output:

  0 1
  1
  0 3
  2
  1 3
  1
  -1
```

Exercise 2. MSB and LSB**(call your program msb_lsb.c)**

Create a C program as follows. First define a function called msb (most significant bit) that takes as input a positive non-zero integer (unsigned int) and returns the position of the most significant bit that is set to 1. Next, write a function called lsb that takes as input a positive non-zero integer (unsigned int) and returns the position of the least significant bit that is set to 1. Note that if the unsigned integer has a single bit set to 1 (or is a power of two) then msb and lsb should return the same position. Your main function asks the user to enter a positive non-zero integer and then outputs the distance between the first and last bits set to 1. For example, if the input is 9 (which is ....1001 in binary) then the output would be 3 because 3 - 0 is 3.

```
Example output:
  1
  0
```

```
Example output:
  9
  3
```

Exercise 3. Arrays(call your program my_arrays.c)

Create a C program, analysis.c. Create an array of size 10 using only the numbers 1, 2 and 3 (yes the numbers will repeat).

1. Define a symbolic constant called SIZE with value 10, to be used to refer to the size of the array when needed.
2. Implement a function printArray that takes a 1D array of integers, and prints it as shown below.

```
Index Value
    0     1
    1     1
    2     1
    3     1
    4     1
    5     2
    6     2
    7     2
    8     3
    9     3
```

3. Implement a function arrayHistogram that takes an array, computes its histogram (the frequency of every value in the array), and draws it as shown:

```
Value Frequency Histogram
    1         5     *****
    2         3       ***
    3         2        **
```

4. Implement a function – swapValues – that takes an array, and two indices. It then swap the values at these indices.
5. Implement a function – bubbleSort – that sorts an array using bubble sort algorithm:
   (a) Several passes will be performed through the array
   (b) On each pass, successive pairs of elements (0 and 1, then 1 and 2 . . .) are compared
   (c) If a pair is in increasing order, we leave the values as they are

(d) If not, the values are swapped in the array (using the function swapValues)

(e) You repeat until no more values are being changed.

6. Implement a function – median – that calculates the median of an array (the value in the middle of the sorted array) (Use the function bubbleSort to sort your array).
7. Implement a function – mode – that returns the mode of an array (the most existing value in the array).
8. Implement a function – isSorted – that takes a 1D array of integers, and its size as arguments, and returns 1 if the array is sorted in ascending order and 0 otherwise.
9. Create a main function to test all the implemented functions.

Exercise 4. More Pointers(call your program my_pointers.c)

Write a C function - merge - that takes an input two sorted arrays (sorted in alpha-numerical order) of strings and the number of strings in each array. It then returns a new array containing all strings in sorted (alpha-numeric) order. In the main function, create two arrays different strings of your choice. Note that, nor the arrays nor the strings have to be of the same length. Call the function reverse. Display the strings returned by the function (one on each line).

```
Example input:
{"ab", "ac"} and {"za", "zb", "zzzzc"}

Example output:
ab
ac
za
zb
zzzzc
```

Exercise 5. More Arrays(call your program more_arrays.c)

Write a C function - concat - that takes a 2D array of strings as input (and the number of strings). It then returns a 1D array containing the concatenation of elements in each element of the 2D array with a space separation between each two elements. For example:

$$\{\{\text{"I"},\text{"LOVE"},\text{"CODING"}\},\{\text{"THIS"},\text{"IS"},\text{"SPARTA"}\}\}$$

$$\rightarrow \{\text{"I LOVE CODING"},\text{"THIS IS SPARTA"}\}$$

.

Faculty of Arts & Sciences
Department of Computer Science
CMPS 270 – Software Construction
Fall 2022 – Assignment 1

In addition to your GitHub submission, add your programs to one folder: username.a01.zip (or .rar) and submit it to moodle.