# Subject: PRF192- PFC
# Workshop 06: ARRAY

**Objectives:**

- Practicing skills, indexing, getting values and implementing the array
- Programs using user-defined functions with various C syntax programming.

## Program 1 (2 marks)

Make a C-program that let user input an 1-D array of integer numbers with size of n elements (maximum of n as 100 elements). After that, find and print out the values and their position of the largest and the second largest elements in this array.

## Program 2 (2 marks)

Write a C-program that let user input an 1-D array of float numbers with size of n elements (maximum of n as 50 elements). After that, make sorting and print out the array of these float elements in both of ascending and descending orders by bubble sort algorithm.

## Program 3 (3 marks)

Develop a C-program that helps user managing an 1-D array of integer numbers (maximum of 100 elements), with initial number of elements is 0, using the following simple menu:

1- Add  a value
2- Search a value
3- Print out the array
4- Print out values in a range
5- Print out the array in descending order
6- Print out the strong numbers
7- Others- Quit

- When the option 1 is selected, user will enters a value then it is added to the array
- When the option 2 is selected, user will enters a value then the position of its existences will be printed out.
- When the option 3 is selected, all values in the array will be printed out.
- When the option 4 is chosen, user will enter 2 values, minVal and maxVal, the values in array which are between minVal and maxVal will be printed out (minVal <=value<=maxVal).
- When the option 5 is chosen, values in array will be printed out in descending order by selection sort algorithm.
- When the option 6 is selected, all strong numbers in the array will be printed out.

**Program 4 (3 marks)**

## ISBN

This problem is slightly more difficult than the above problem. This one requires a data type that stores 10 digits. Please use array to store these 10 digits.

## Background

Publishers and bookstores use a number system called the International Standard Book Number (ISBN) system to identify books. At the start of publication, each book is assigned a unique ISBN. An ISBN, once assigned, can never be re-used.

An ISBN consists of exactly 10 digits. The rightmost digit is the check digit. The check digit is validated modulo 11.

- multiply each digit from the first to the ninth by a weight from 10 to 2 respectively (the first digit by 10, the second by 9,..., the ninth by 2).
- the sum of the products plus the check digit should be divisible without remainder by 11.
- if there is a remainder, the whole number is not a valid ISBN.

Consider the following example:

```
ISBN   0003194876
                |
check digit is 6
add first set of alternates to themselves
   0   0   0   3   1   9   4   8   7
  10   9   8   7   6   5   4   3   2
   0   0   0 21   6 45 16 24 14 = 126
 add check digit                    6
 total                            132
 divide by 11                      12
 remainder                          0
 Therefore this ISBN is valid
```

## Specifications

Design a program that validates an ISBN. Your program keeps accepting a whole number and determining if that whole number is a valid ISBN. Your program terminates when the user enters 0 as the whole number.

The output from your program looks something like:

```
ISBN Validator
==============
 ISBN (0 to quit): 0003194876
 This is a valid ISBN.
 ISBN (0 to quit): 0003194875
 This is not a valid ISBN.
 ISBN (0 to quit): 0
 Have a Nice Day!
```

The data type **long** only guarantees room for 9 digits. The data type **long long** guarantees room for an integer with well over 12 digits (at least 64 bits of precision). The conversion specifier for a **long long** integer is **%lld**.