Assignment 5

**Evaluation weight:**

5%

THIS ASSIGNMENT HAS BEEN MODIFIED. AND MORE TIME HAS GIVEN TO YOU !

**Submission specifications:**

* Name the program correctly: pw2Bank + *name1* + *name2* [ + *name3* ]
* The code of the program and any dependencies
* Use of Array is MANDATORY

**Description:**

Create a prototype of a program that simulates a bank. More precisely, the program should be able to store the family name and given name of a client, their unique account number (a number between 10000 and 10099), and the balance of their account.

The program should display a menu with the following options:

1. Add a bank account.
2. Remove a bank account. (always a shift back is mandatory)
3. Display the information of a particular client’s account. (by account number)
4. Apply a deposit to a particular account. (by account number)
5. Apply a withdrawal from a particular account. (by account number)
6. Sort and display the list of clients according to their balance, family name and given name, in ascending or descending order.
7. Display the average balance value of the accounts.
8. Display the total balance value of the accounts.
9. You should be capable to create an account and insert it at any position of the Array.

This implies shifting back or forward the other elements of the Array.

1. Exit the application.

You must implement each of these functionalities

**Remarks:**

* The maximum number of accounts for this application is 100 accounts.
* All input must be validated. **(type, length , no duplicate)**
* Each task out of the various functionalities should be delegated to an appropriate function.

**Evaluation:**

The project will be graded according to the following criteria:

* **Exactitude of the program:**Does the program do what it is supposed to do?
* **Visual appearance:**All text displayed to the screen should be well arranged and written in proper English.
* **Input validation:**The program should properly handle errors that can occur during data input.
* **Structure of the code:**The program should be split up into functions according to the relevant needs.
* **Arrangement and clarity of the code:**The code should be properly indented, with relevant comments where needed, and it should respect programming conventions (variables in camelCase, constants in UPPER\_CASE\_SEPARATED\_BY\_UNDERSCORES). Hard-coded values should be put into constants.