Object Oriented Programming Lab

Lab 10 **Marks 10**

Instructions

Work on this lab individually. You can use your books, notes, handouts etc. but you are not allowed to borrow anything from your peer student. *You are strictly **NOT ALLOWED** to include any additional data-members/functions/constructors* in your class.

Marking Criteria

Show your work to the instructor before leaving the lab to get some or full credit.

What you must do

Program the following task in your C++ compiler and then compile and execute them. *Write the **main** function first and keep testing the functionality of each function once created.*

ADT: Generic Array

Write a **generic class** named **Array** that can hold an array of any valid data type *(int, float, char etc…)*.

1. The class should have following **two private data members**.
    1. A **generic pointer** named **data** that holds an **array of given data type** allocated dynamically according to the specified **size**.
    2. An **integer** named **size** that holds the **size of the array** *(amount of memory allocated to data)*.

2. Provide the implementation of following **constructors** and a **destructor**
    1. A **default constructor** who allocates an array of **size 5** and initializes it to the so-called "empty array," i.e., an array whose array representation **contains all zeroes**.
    2. A **constructor** who accepts an **integer** as argument to represent the **size of an array** and initializes it to the so-called "empty array," i.e., an array whose array representation **contains all zeroes**.
    3. A **copy constructor** to initialize an array object with already existing object.
    4. A **destructor** to **free any memory resources** occupied by the **array** object.

3. Provide the implementation of following member functions and operators
    1. **getSize** returns the size of array.
    2. **setElement** that **inserts** a new element **k** at index **i** (both passed as argument) into an **array**, if possible, otherwise give an appropriate error message.
    3. **countElement** accepts a **key** as argument and **count and return** the **total occurrences** of it in an array, **-1** otherwise, if the **key does not exist**.
    4. **Assignment (=)** which copies the data of *right-hand-side* object to the *left-hand-side* object. If the size of *left-hand-side* object is different from *right-hand-side* object. Reallocate the memory to *left-hand-side* object according to the size of *right-hand-side* object and then copy the data. Don't forget to update the **size** of *left-hand-side* object, if required.
    5. **getSubArray** receives two integer parameters as argument **start_index** and **end_index** and **return** a **new Array** which contains all the values exist in the *left-hand-side* object from **start_index** to **end_index** both inclusive, if possible.
    6. **Arithmetic binary (+)** that insert contents of *right-hand-side* object at the end of *left-hand-side* object and **return** the result. The function should not make any changes in *left/right-hand-side* object.
    7. **Stream insertion (<<)** to take input from user for the **data** of an **array**.
    8. **Stream extraction (>>)** to display the contents of **data** on the screen of an **array**.
    9. **Comparison (==)** that determines whether **two arrays are equal or not**. The operator should return **true** if both the arrays *(left-hand-side and right-hand-side)* are equal, **false** otherwise.
    10. **Subscript ([])** for both lvalue and rvalue of non-const objects.
    11. **Subscript ([])** for rvalue of const objects.

4. Once you have written the class, write **main** function and test its functionality by creating some objects of **Array**.

☺ ☺ ☺ **BEST OF LUCK** ☺ ☺ ☺