Object Oriented Programming Lab

Lab 09 **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: Collection

Write a class named **Collection** for which each object can hold **negative integers** and **zero** as a default value.

1. The class should have following **two private data members**.
    1. An **integer pointer** named **data** that holds an **array of integers** 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 **constructor** which accepts an **integer** as argument to represent the **size of an array** and initializes it to the so-called "empty collection," i.e., a collection whose array representation **contains all zeroes**.
    2. An additional **constructor** that receives an **array of integers** and the **size of that array** as its arguments and uses the array to initialize a **collection object**.
    3. A **copy constructor** to initialize a collection object with already existing object.
    4. A **destructor** to **free any memory resources** occupied by the **collection** object.

3. Provide the implementation of following member functions and operators
    1. **getSize** returns the size of collection.
    2. **setElement** that **inserts** a new integer **k** at index **i** *(both passed as argument)* into a **collection**, if possible, otherwise give an appropriate error message.
    3. **countElement** accepts an integer **key** as argument and **count and return** the **total occurrences** of it in a collection, **-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.
    5. **getSubCollection** receives two integer parameters as argument **start_index** and **end_index** and **return** a **new Collection** which contains all the values exist in the left-hand side object from **start_index** to **end_index** both inclusive, if possible.
    6. **Stream insertion (<<)** to take input from user for the **data** of a **collection**.
    7. **Stream extraction (>>)** to display the contents of **data** on the screen of a **collection**.
    8. **Arithmetic assignment (+) binary** which perform the addition of two collections (left-hand side and right-hand side) if possible and **return** the result.
    9. **Comparison (==)** that determines whether **two collections are equal or not**. The operator should returns **true** if both the collections are equal, **false** otherwise.
    10. **Subscript ([])** for both lvalue and rvalue of non-const objects.
    11. **Subscript ([])** for rvalue of const objects.
    12. **Unary minus (-)** return **true** if all the elements of a collection are **non-zeroes**, **false** otherwise.

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

☺ ☺ ☺ **BEST OF LUCK** ☺ ☺ ☺