11 - Programming Exercises
IntegerSet ADT

Program the following task in your C++ compiler. Keep compiling and executing even after writing a single line of code.

ADT: IntegerSet

Create a class named **IntegerSet** that represent a set of integers in the range from **0** to **Size - 1**, where **size** is a constant integer representing the maximum capacity of the set. Internally, the set should be represented as an array of **ones** and **zeros**, where array element **a[k]** is 1 if integer **k** is in the set, and 0 if integer **k** is not in the set.

For Example, the following set contains values 0, 1, 3, 4, 7 and 9.

0	1	2	3	4	5	6	7	8	9
1	1	0	1	1	0	0	1	0	1

1. The class should have the following private data members.

- An integer pointer to hold a dynamically allocated array representing the set.
- A constant integer to store the maximum size of the array.

2. Provide the implementation of following constructors and a destructor.

- A constructor that accepts an integer representing the size of the set and initializes it as an empty set (all elements set to 0).
- A copy constructor to initialize a set object with an existing object.
- A destructor to release any dynamically allocated memory.

3. Operators to be overloaded:

- **Stream insertion operator (<<)** to print the set as a list of numbers separated by spaces, printing only the elements present in the set. If the set is empty, print "---".
- **Assignment operator (=)** to copy the data of one object to another, avoiding self-assignment. The copy should only occur if both objects have the same size.
- **Equality operator (==)** to determine whether two sets are equal. Return **true** if both sets are equal, **false** otherwise.
- **Logical NOT operator (!)** to create and return a new set containing the reverse of the left-hand side object.

4. Member functions for common set operations:

- **insertElement(int k):** Inserts integer **k** into the set by setting **a[k]** to 1.
- **deleteElement(int k):** Deletes integer **k** from the set by setting **a[k]** to 0.
- **unionOfSets(const IntegerSet& set1, const IntegerSet& set2):** Creates a third set that is the union of two existing sets.
- **intersectionOfSets(const IntegerSet& set1, const IntegerSet& set2):** Creates a third set that is the intersection of two existing sets.
- **findElement(int key):** Searches for integer key in the set and returns **true** if found, **false** otherwise.
- **isNullSet():** Returns **true** if the set is an empty set, **false** otherwise.

5. Guidelines:

- Ensure proper memory management throughout the class implementation.
- Implement appropriate error handling for invalid input and operations on sets of different sizes.
- Test the class thoroughly with various scenarios to ensure correctness and robustness of the implementation.

☺ ☺ ☺ **If you can dream it, you can do it.** ☺ ☺ ☺