Homework:

Write a code to perform the following steps:

1. Create a class for a 1D Array, used to store integers; it should have the following members and attributes (you can add more as needed)

a. int *num*, store the number of integers in this array.

b. int* *data*, a pointer to a memory block to store the integers.

c. int *cap*, the size of the memory block, which is equal or greater than *num.* A constraint on your task is that *cap* should always be a power of 2.

d. - A default constructor which creates an empty vector,
- a copy constructor, and
- an assignment operator.

e. A non-default constructor to set the size of the vector. Note that the capacity should always be a power of 2.

e. A destructor to free any allocated memory.

f. Member function, push_back(int i); which adds integer "i" to the end of the array. You may need to increase the memory block size (the capacity) to store this integer. A convenient strategy is to double the capacity when memory is not sufficient.

g. Member function, pop_back(), which removes the last element.

h. Member function, remove(int i), which removes the integer with index i from the array. You need to move the integers that follow the removed integer towards the front of the array.

i. Member function insert(int num, int i), which inserts an integer "num" at the position "i". This also may have an insufficient capacity problem.

j. Member function, capacity(), which returns the capacity of the vector.

k. Member function, size(), which returns the number of elements of the vector.

l. Member function, clear(), which removes all the elements from the array.

2. Create a main function to test your class. *All member functions* must be tested.

3. Create a Makefile to compile and link your code.

Finally, and for 30% of the score, create tests that verify that each function is operating properly. The functions to test are:

- Constructor, copy constructor, assignment operator, destructor
- push_back, pop_back, remove, insert, capacity, size, clear.

For each function, create two tests (you can use the same function with different values).
For each test, you will create a scenario for which you know the outcome, run the test, and verify that the outcome is what you expected.
For example, if you write a function to sum two floats, and want to test it, consider the floats 3.14 and 5.27. The sum should be 8.41, and therefore, the test would check that

$$| \text{sum}(3.15, 5.27) - 8.41 | < 1.e\text{-}7$$

Note: the equality might not be exact when working with floats, so you allow for an error tolerance. This would not be required if working with integers.

Put all your tests in a single file, which is linked against *main_test.cpp*, and create an executable called *test.x* . When running *test.x*, return Passed/Failed for each of your tests.

Zip your code, with a readme file (txt or pdf) and submit to Canvas.