POO (17:00 - 18:00)

Write the class **PriorityQueue** (that contains a fixed-sized string) so that the following code

```
int main()
{
    PriorityQueue p(6);
    (((((((p += 5) += 2) += 8) += 3) += 10) += 1) += 11) += 9;
    cout << "Count = " << (int)p << endl;
    while ((int)p) {
        cout << (p--) << ",";
    }
    cout << endl;
    (((p += 1) += 2) += 3) += 1;
    cout << "Sum = " << p("sum") << endl;
    cout << "Count = " << (int)p << endl;
    cout << "Min = " << p("min") << endl;
    cout << "Max = " << p("max") << endl;
}
```

compiles and upon execution prints the following to the screen:

```
Count = 6
11,10,9,8,5,3,
Sum = 7
Count = 4
Min = 1
Max = 3
```

Carefully read the main function to deduce what methods/operators should be included in PriorityQueue class.

Constraints:

- You are not allowed to use **STL** at all (for vectors, strings, maps or any **template/object** defined in STL). The only exception is the usage of "**std::cout**" from the main function
- You are not allowed to use string manipulation functions defined in "string.h" such as **strlen**, **strcpy**, **strdup, strtok, strcmp**, etc, or string to number conversions.
- You are not allowed to use any preexisting sort method (you will have to write one for yourself). Performance is not relevant for this test !
- If you don't respect the previous conditions (e.g. use strlen, or strcpy, etc) → we will compute the correctness of the code, but the final grade will be half of the computed score for each particular code that uses those functions.

Observations:

- The constructor has as input the maximum number of elements in the queue.
- The decrement operator works like a pop method (removing the largest number from the priority queue).

Grading (informative):

G1	**PriorityQueue** destructor	1p
G2	**PriorityQueue** constructor	2p
G3	Organize your project in 3 files: **main.cpp**, **PriorityQueue.h** and **PriorityQueue.cpp**	1p
G4	Organize your class **PriorityQueue**to include private and public members, the definition of a constructor, and at least one operator. The private data should include one pointer and information about the size of the priority queue and allocated memory.	2p
G5	Operator += that adds an element to the queue and then sorts the queue	7p
G6	Operator-- that removes the first elements from the queue	5p
G7	Cast to int type that provides the number of elements from the queue	1p
G8	Proper format for operator function call	2p
G9	"**sum**" support for operator function call	2p
G10	"**min**" support for operator function call	2p
G11	"**max**" support for operator function call	2p
G12	The program compiles and upon execution produces the expected results	3p