POO LAB 2

Write the class **Matrix ,** whose objects are bidimensional arrays arranged in row and column so that the following code

```
int main() {
    Matrix m(4, 3);
    m.fill(5); m(0, 0) = 30; m(3, 2) = 20;
    m.print();
    printf("W:%d,H:%d,Size:%d\n", m["width"], m["height"], m["size"]);
    Matrix s(3, 3);
    if (s.is_square_matrix()) { printf("Square\n"); }
    s.fill(0);
    s(0, 0) = 1; s(1, 1) = 1; s(2, 2) = 1;
    if (s.is_identity_matrix()) { printf("Identity\n"); }
    s.print();
    if (m == s) { printf("Egale"); } else { printf("diferite"); }
}
```

compiles and upon execution prints the following to the screen:

```
30  5  5  5
 5  5  5  5
 5  5  5 20
W:4,H:3,Size:48
Square
Identity
 1  0  0
 0  1  0
 0  0  1
diferite
```

Observations:

- 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.
- 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 particula code that uses those functions.
- For the purpose of this example, we will consider that all indexes are correct (within the limit of the initialized matrix) and the size of the matrix (width and height) upon initialization is correct (bigger than 0).
- The data where the matrix values are stored, MUST be dynamically allocated (via new operator). Using a fixed size buffer/matrix will NOT be considered a valid solution.
- The "size" of a matrix is its width multiplied with its height and the size of an int (4 bytes).

Grading (informative):

G1	Matrix constructor with two integer values	3p
G2	Matrix destructor (that deletes the heap allocated buffer for matrix values)	1p
G3	Organize your project in 3 files: **main.cpp**, **Matrix.h** and **Matrix.cpp**	1p
G4	Organize your class **Matrix** to include private and public members, the definition of a constructor and destructor, and at least one method.	2p
G5	Operator() to access an element from the matrix	5p
G6	Operator[] to get informations such as width, height or size	4p
G7	Operator== to check if two matrices are equal	3p
G8	Method: print that will print the matrix	2p
G9	Method: is_identiy_matrix that true if the matrix is the Identity matrix	3p
G10	Method: is_square that returns true if the matrix width and height are equal	1p
G11	Method: fill that fills the matrix with a specified value	2p
G12	Program compiles and upon execution produces the expected results	3p