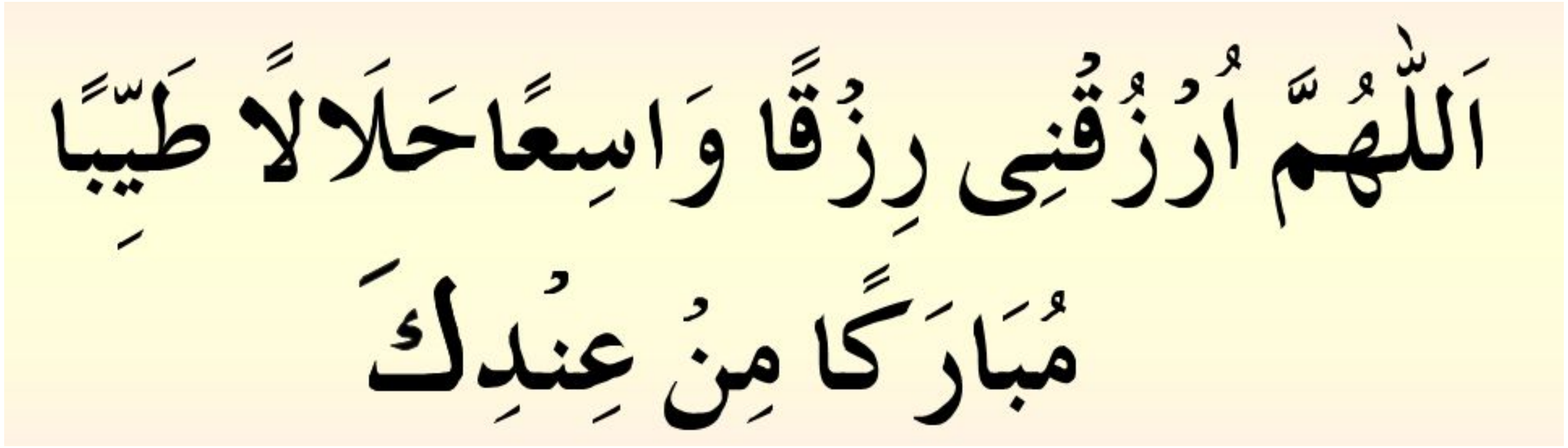
2D Arrays (Complex Problem)

اَللّٰهُمَّ اُرُزُقُنِى عِلُمًا نَافِعًا وَاسِعًا عَمِيُقًا



اَللّٰهُمَّ اُرْزُقْنِى رِزُقًا وَاسِعًا حَلَالًا طَيِّبًا
مُبَارَكًا مِنُ عِنُدِكَ

Working Example: Gravity

Given a 2D array (5x5) of some suspended blocks (represented as hashtags),create three functions:

1. DisplayWorld()
2. SetGravityStatus(bool status)
3. TimeTick(int times)

At the moment, we are assuming last row is hard ground and the block stay on it when it hit at ground. Other blocks can pile on each other.

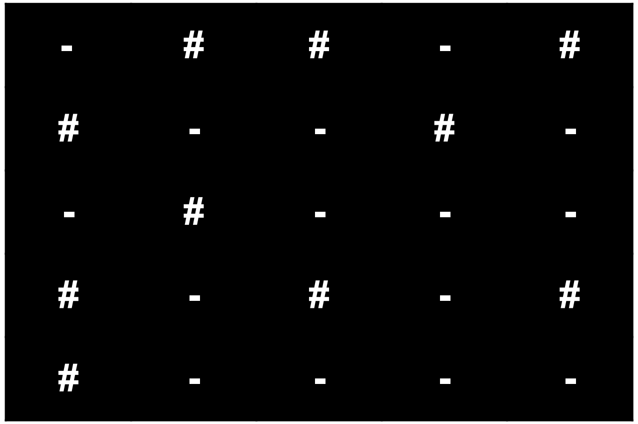
-	#	#	-	#
#	-	-	#	-
-	#	-	-	-
#	-	#	-	#
#	-	-	-	-

Working Example: displayWorld()

```
char objects[5][5] = {
    {'-', '#', '#', '-', '#'},
    {'#', '-', '-', '#', '-'},
    {'-', '#', '-', '-', '-'},
    {'#', '-', '#', '-', '#'},
    {'#', '-', '-', '-', '-'}
    };

main()
{
    displayWorld();

}
```



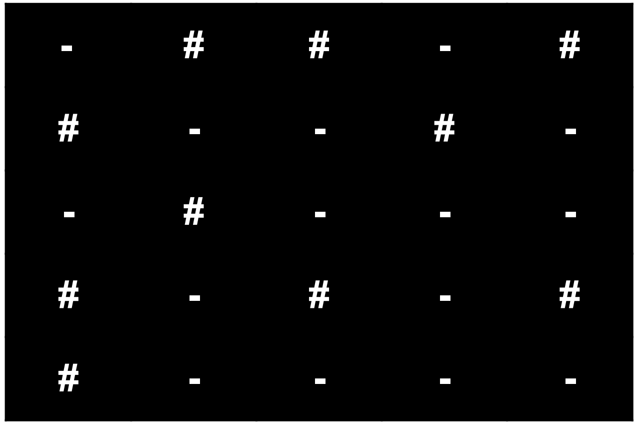
```
-    #    #    -    #
#    -    -    #    -
-    #    -    -    -
#    -    #    -    #
#    -    -    -    -
```

displayWorld() will show the 2D array on the screen.

Working Example: setGravityStatus(true)

```
char objects[5][5] = {
    {'-', '#', '#', '-', '#'},
    {'#', '-', '-', '#', '-'},
    {'-', '#', '-', '-', '-'},
    {'#', '-', '#', '-', '#'},
    {'#', '-', '-', '-', '-'}
    };
bool gravity = false;
main()
{
    displayWorld();
    setGravityStatus(true);

}
```

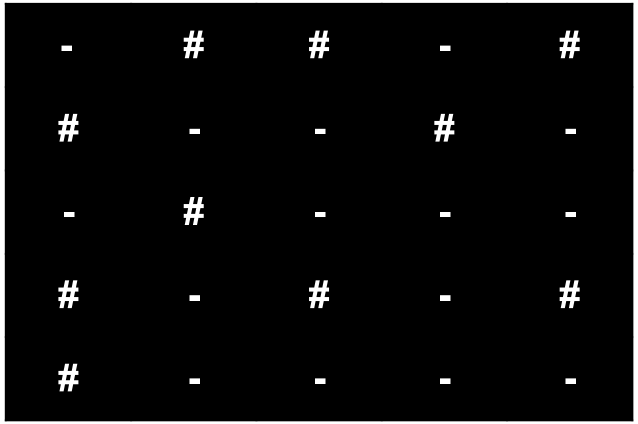


```
-    #    #    -    #
#    -    -    #    -
-    #    -    -    -
#    -    #    -    #
#    -    -    -    -
```

setGravityStatus() function change the gravity of world but will not make any changes in the array or the display

Working Example: timeTick(1)

```
char objects[5][5] = {
    {'-', '#', '#', '-', '#'},
    {'#', '-', '-', '#', '-'},
    {'-', '#', '-', '-', '-'},
    {'#', '-', '#', '-', '#'},
    {'#', '-', '-', '-', '-'}
    };
bool gravity = false;
main(){
    displayWorld();
    setGravityStatus(true);
    timeTick(1);
    displayWorld();
}
```



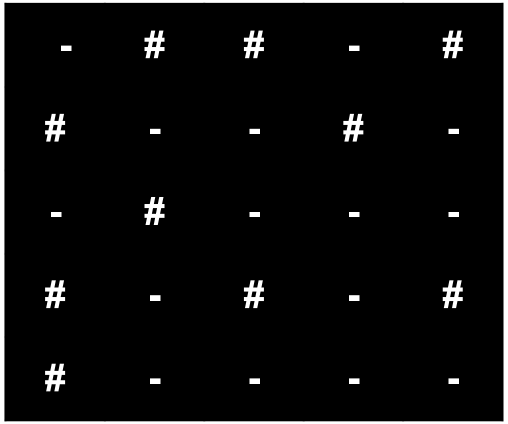
```
-    #    #    -    #
#    -    -    #    -
-    #    -    -    -
#    -    #    -    #
#    -    -    -    -
```

TimeTick(1) will change the world by one tick. We are assuming after tick all block fall one level down if possible.

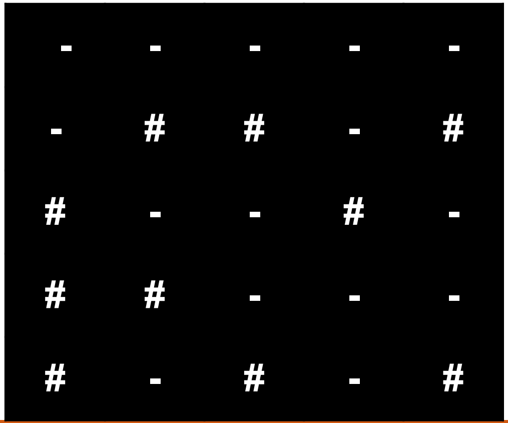
Working Example: timeTick(1)

```
char objects[5][5] = {
    {'-', '#', '#', '-', '#'},
    {'#', '-', '-', '#', '-'},
    {'-', '#', '-', '-', '-'},
    {'#', '-', '#', '-', '#'},
    {'#', '-', '-', '-', '-'}
    };
bool gravity = false;

main(){
    displayWorld();
    setGravityStatus(true);
    timeTick(1);
    displayWorld();
}
```



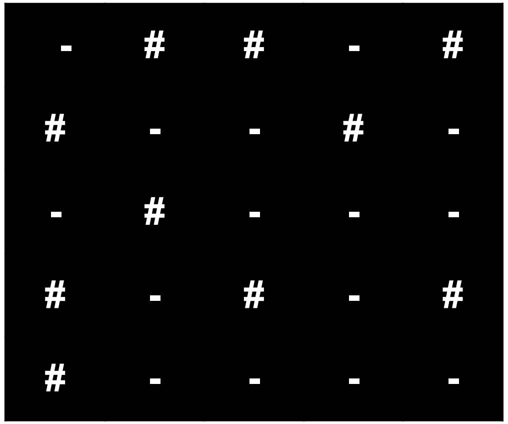
```
-  #  #  -  #
#  -  -  #  -
-  #  -  -  -
#  -  #  -  #
#  -  -  -  -
```



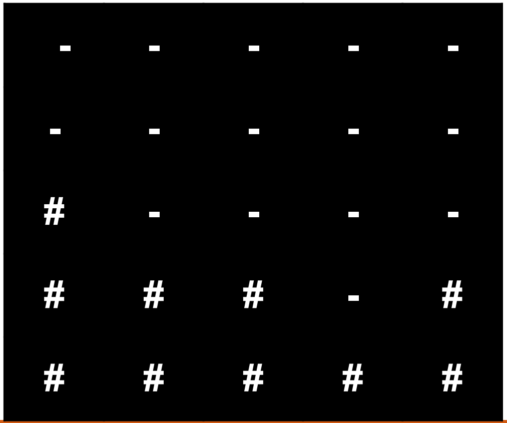
```
-  -  -  -  -
-  #  #  -  #
#  -  -  #  -
#  #  -  -  -
#  -  #  -  #
```

Working Example: timeTick(3)

```
char objects[5][5] = {
    {'-', '#', '#', '-', '#'},
    {'#', '-', '-', '#', '-'},
    {'-', '#', '-', '-', '-'},
    {'#', '-', '#', '-', '#'},
    {'#', '-', '-', '-', '-'}
    };
bool gravity = false;
main(){
    displayWorld();
    setGravityStatus(true);
    timeTick(3);
    displayWorld();
}
```



```
-  #  #  -  #
#  -  -  #  -
-  #  -  -  -
#  -  #  -  #
#  -  -  -  -
```

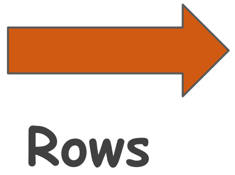


```
-  -  -  -  -
-  -  -  -  -
#  -  -  -  -
#  #  #  -  #
#  #  #  #  #
```

Working Example

-	#	#	-	#
#	-	-	#	-
-	#	-	-	-
#	-	#	-	#
#	-	-	-	-

Working Example



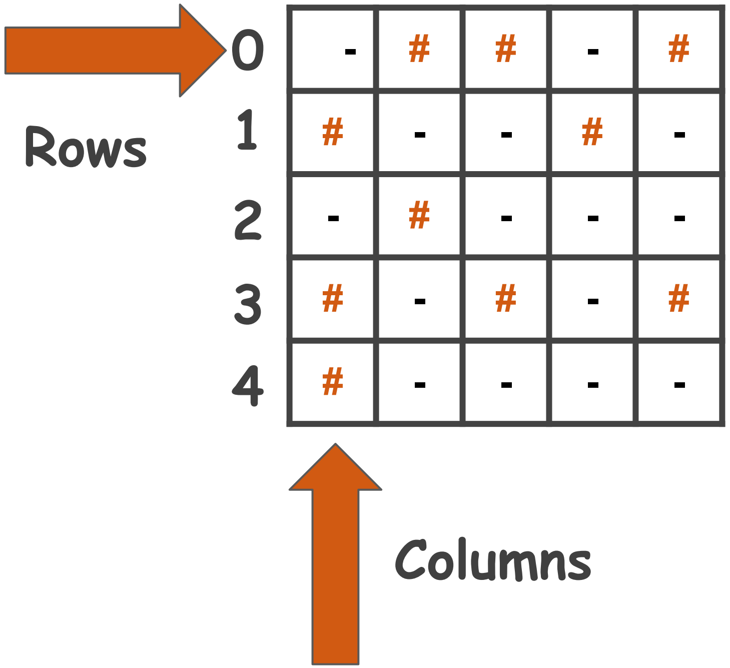
-	#	#	-	#
#	-	-	#	-
-	#	-	-	-
#	-	#	-	#
#	-	-	-	-

Working Example

Rows

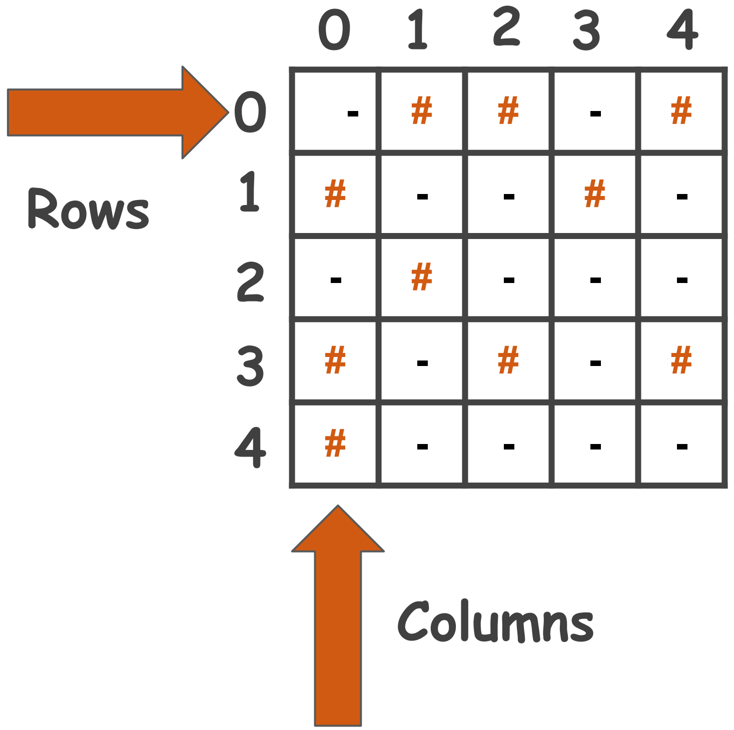
0	-	#	#	-	#
1	#	-	-	#	-
2	-	#	-	-	-
3	#	-	#	-	#
4	#	-	-	-	-

Working Example



0	-	#	#	-	#
1	#	-	-	#	-
2	-	#	-	-	-
3	#	-	#	-	#
4	#	-	-	-	-

Working Example



	0	1	2	3	4
0	-	#	#	-	#
1	#	-	-	#	-
2	-	#	-	-	-
3	#	-	#	-	#
4	#	-	-	-	-

displayWorld()

Make the function to display all the contents of 2D Array.

	0	1	2	3	4
0	-	#	#	-	#
1	#	-	-	#	-
2	-	#	-	-	-
3	#	-	#	-	#
4	#	-	-	-	-

```
void displayWorld()
{
    for (int row = 0; row < 5; row++)
    {
        for (int col = 0; col < 5; col++)
        {
            cout << objects[row][col];
            cout << "\t";
        }
        cout << endl;
    }
    cout << endl;
}
```

setGravityStatus(bool value)

Make the function to change the gravity status according to the parameter passed

	0	1	2	3	4
0	-	#	#	-	#
1	#	-	-	#	-
2	-	#	-	-	-
3	#	-	#	-	#
4	#	-	-	-	-

```
void setGravityStatus(bool value)
{
    gravity = value;
}
```

timeTick()

Make the function to move the **buildings** down according to the time ticks.

```
void timeTick(int times)
{
    int count = 0;
    if (gravity)
    {
        while (count < times)
        {
            for (int row = 3; row >= 0; row--)
            {
                for (int col = 0; col < 5; col++)
                {
                    if (objects[row][col] == '#')
                    {
                        if (objects[row + 1][col] == '-')
                        {
                            objects[row + 1][col] = '#';
                            objects[row][col] = '-';
                        }
                    }
                }
            }
            count = count + 1;
        }
    }
}
```

Learning Objective

Write C++ Program that solves complex real world problem using 2D arrays.

Self Assessment: Video Profile Activity

Now, you need to change the block activity but assume the last row is not solid ground instead that is black hole that port everything back to first row when it tries to get out of it. The last row will only act as black whole if the flag variable isBlackHole set to true. In case, its value is false the behaviour of the program will remain same as it was previously with solid ground.

Working Example: timeTick(1)

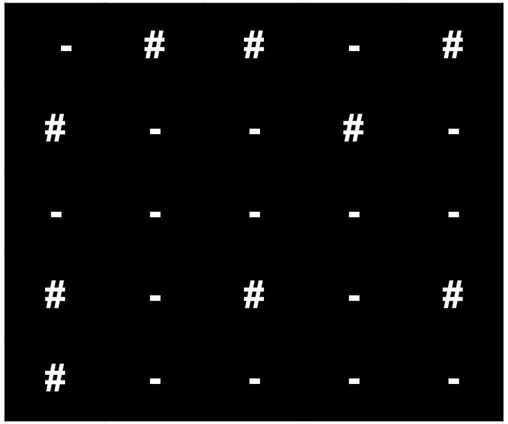
```
char objects[5][5] = {
    {'-', '#', '#', '-', '#'},
    {'#', '-', '-', '#', '-'},
    {'-', '#', '-', '-', '-'},
    {'#', '-', '#', '-', '#'},
    {'#', '-', '-', '-', '-'}
    };
bool gravity = false;
bool isBlackHole = true;
main(){
    displayWorld();
    setGravityStatus(true);
    timeTick(1);
    displayWorld();
}
```

```
-  #  #  -  #
#  -  -  #  -
-  -  -  -  -
#  -  #  -  #
#  -  -  -  -
```

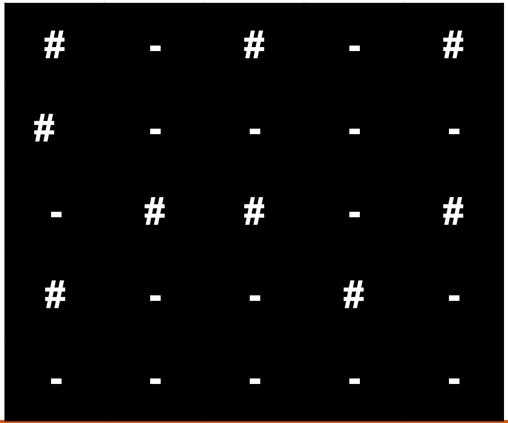
```
#  -  -  -  -
-  #  #  -  #
#  -  -  #  -
-  -  -  -  -
#  -  #  -  #
```

Working Example:timeTick(2)

```
char objects[5][5] = {
    {'-', '#', '#', '-', '#'},
    {'#', '-', '-', '#', '-'},
    {'-', '#', '-', '-', '-'},
    {'#', '-', '#', '-', '#'},
    {'#', '-', '-', '-', '-'}
    };
bool gravity = false;
bool isBlackHole = true;
main(){
    displayWorld();
    setGravityStatus(true);
    timeTick(2);
    displayWorld();
}
```



```
-  #  #  -  #
#  -  -  #  -
-  -  -  -  -
#  -  #  -  #
#  -  -  -  -
```



```
#  -  #  -  #
#  -  -  -  -
-  #  #  -  #
#  -  -  #  -
-  -  -  -  -
```

Self Assessment 02

In probability theory, a probability matrix is a matrix such that:
The matrix is a square matrix (same number of rows as columns). All entries are probabilities, i.e. numbers between 0 and 1. All rows add up to 1.
The following is an example of a probability matrix:

```
[
  [0.5, 0.5, 0.0],
  [0.2, 0.5, 0.3],
  [0.1, 0.2, 0.7]
]
```

Self Assessment 02

Note that though all rows add up to 1, there is no restriction on the columns, which may or may not add up to 1.

Write a function that determines if a matrix is a probability matrix or not.

Fun fact:
for most probability matrices M (for example, if M has no zero entries), the matrix powers M^n converge (as n increases) to a matrix where all rows are identical.

Self Assessment 02

Test Cases:

Input	Output	Explanation
[[0.5, 0.5, 0.0], [0.2, 0.5, 0.3], [0.1, 0.2, 0.7]] isProbMatrix()	true	
[[0.5, 0.5, 0.0], [0.2, 0.5, 0.3]] isProbMatrix()	false	// Not a square matrix.

Self Assessment 02

Test Cases:

Input	Output	Explanation

[
  [0.5, 0.4],
  [0.5, 0.6]
]
isProbMatrix()
	false	// Rows do not add to 1.

[
  [2, -1],
  [-1, 2]
]
isProbMatrix()
	false	// Entries not between 0 and 1.