Imagine a grid that looked like below, where the first row and first column were the fibonacci sequence.
If you wanted to fill in the remaining grid with the following rule:
(The value of each cell is equal to the sum of the number above, to the left, and to the upper left corner of the cell).

Write the pseudocode that fills in the following table below (with loops not magic numbers)

1	1	2	3	5	8	13	21
1	1	2	3	5	8	13	21
2	2	4	6	10	16	26	42
3	3	6	9	15	24	39	63
5	5	10	15	25	40	65	105
8	8	16	24	40	64	104	168
13	13	26	39	65	104	169	273
21	21	42	63	105	168	273	441

2. In last week's lab, you were to provide a list of operators and attempt to memorize your partner's operators.
This week, without looking at your old assignment, are you able to remember what operators you had decided on?

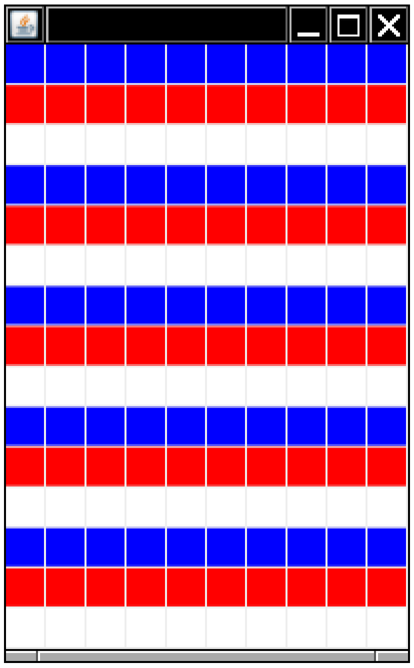
No, we don't remember. I think we have remembered that we used a + operator and - operator, but the numbers we forgot.

Driver: Alan Xiao
Scribe: Froilan Zarate

Two-dimensional arrays.

Scribe: What happened?
All of the cells in the array became green.

Provide pseudocode for creating this pattern:



Pseudocode:

```
from the first row to the last row:
        The row position divided by 3
                If the row's remainder is 0, color the row blue.
                If the row's remainder is 1, color the row red.
                If the row's remainder is 2, color the row white.
```

Ask yourselves: (**Scribe: Record your answers**)

How would you make one row blue

- Color row 0 blue; use a for loop and start at int i = 1.

How do you make all the blue rows?

- For all the rows, for all the columns in that row, color it blue.

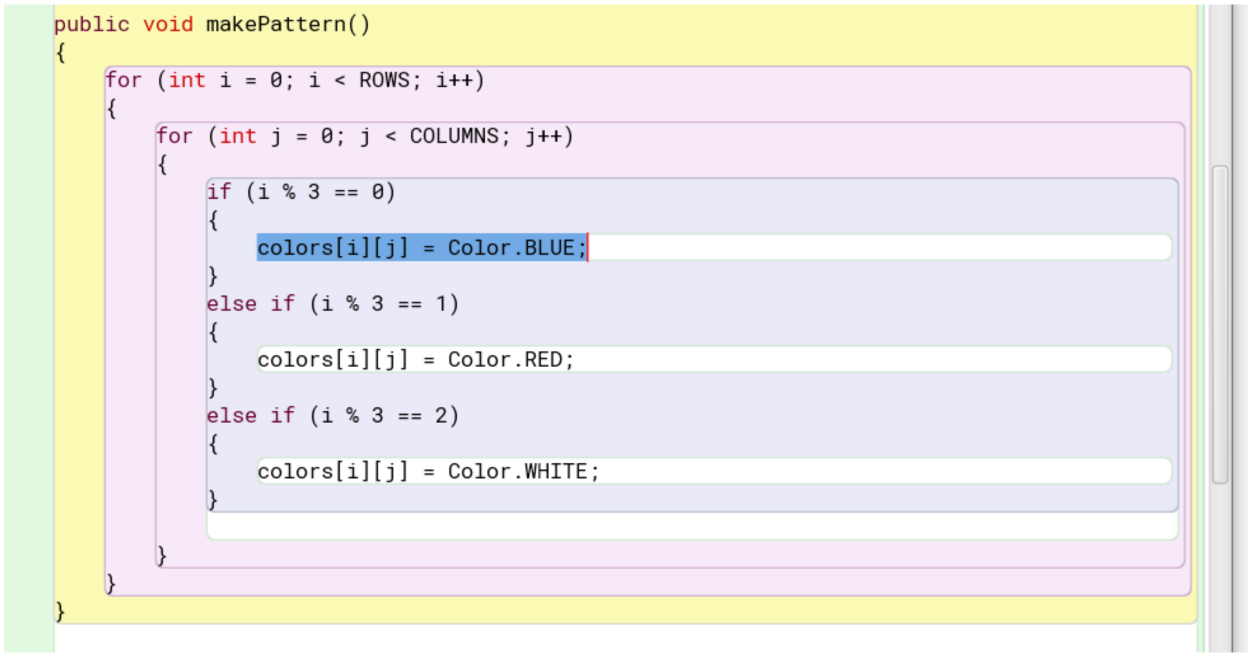
At what index values do you make all the squares in a row blue? (think %)

What if statement could you use to determine what color the row should be?

- 0 (mod 3). i %= 0;

Implement the code and execute it.

Driver: Put the code of the makePattern method in your lab report.

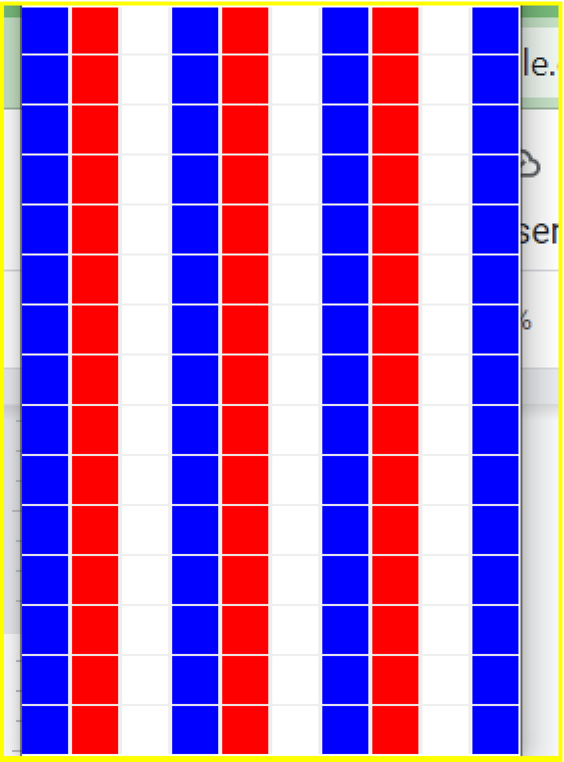


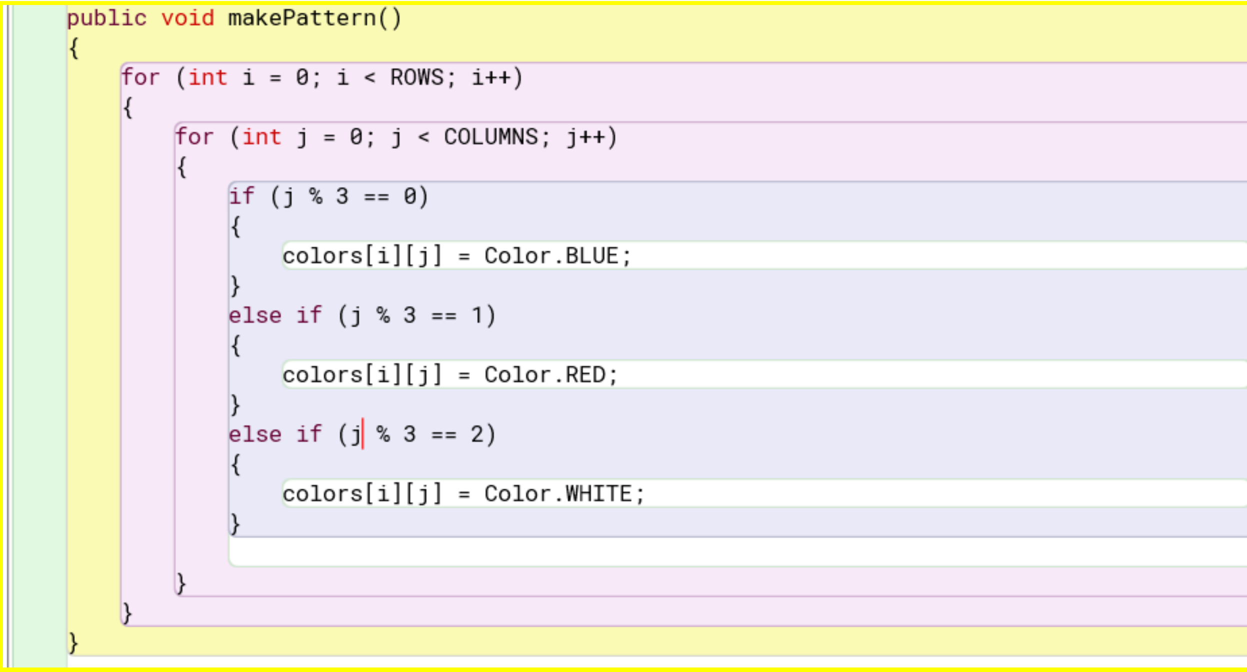
```
public void makePattern()
{
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLUMNS; j++)
        {
            if (i % 3 == 0)
            {
                colors[i][j] = Color.BLUE;
            }
            else if (i % 3 == 1)
            {
                colors[i][j] = Color.RED;
            }
            else if (i % 3 == 2)
            {
                colors[i][j] = Color.WHITE;
            }

        }
    }
}
```

Each of you comes up with a pattern that you want your partner to implement. Sketch the pattern on a sheet of paper. Be creative, but keep in mind that the pattern should be regular enough that it can be done with a couple of loops.
If either of you feels that the pattern you got from the other is too hard to code, you can refuse it. In that case, each of you implements the pattern that you designed. So be sure you can implement your own design.
Discuss the pattern that you got from your partner and be sure you know what your partner wants. Ask for clarification if necessary.

Implement the pattern. **Scribe and Driver: Upload a screen capture along with your report. Paste the code to your lab report.**

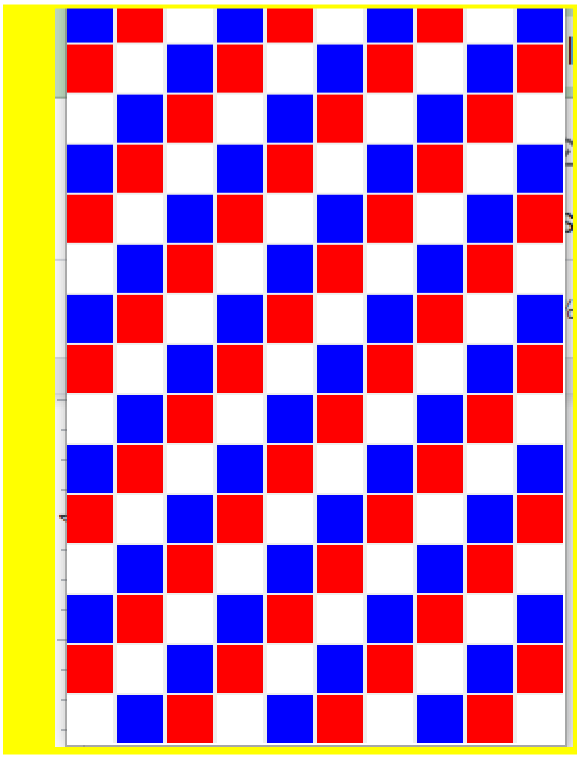
Froilan's Pattern:

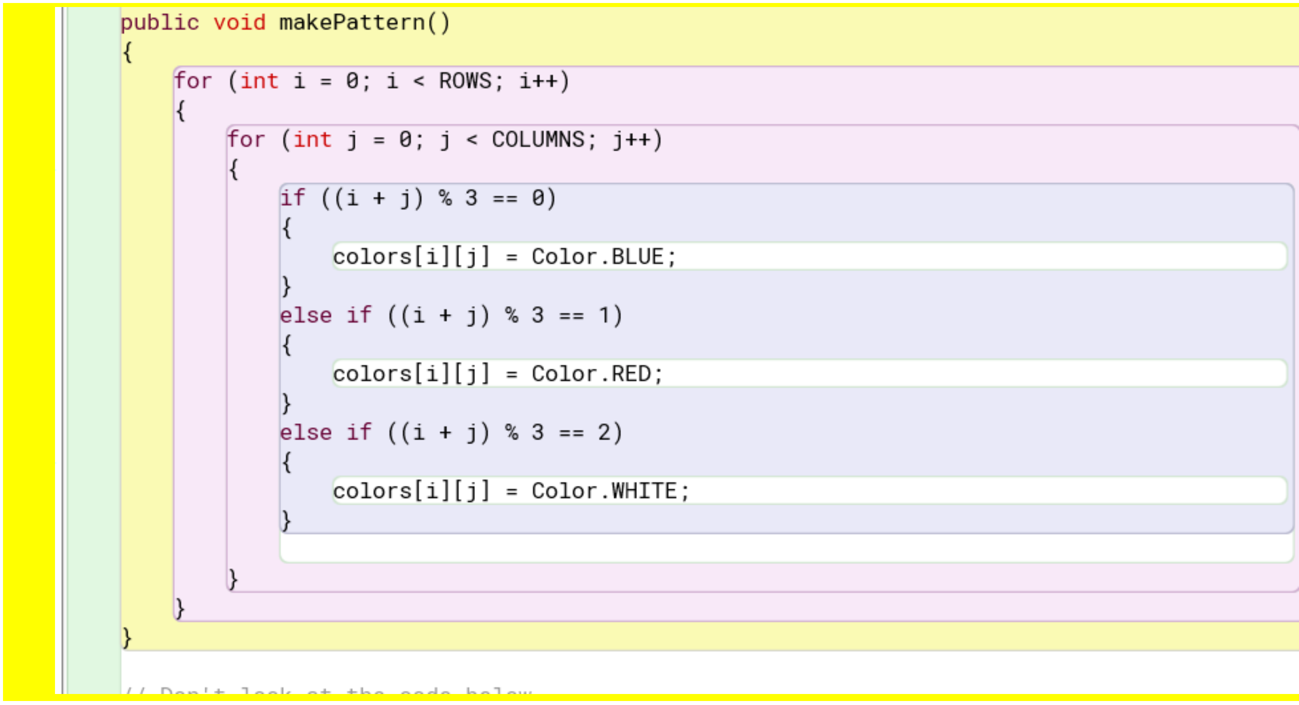




```java
public void makePattern()
{
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLUMNS; j++)
        {
            if (j % 3 == 0)
            {
                colors[i][j] = Color.BLUE;
            }
            else if (j % 3 == 1)
            {
                colors[i][j] = Color.RED;
            }
            else if (j % 3 == 2)
            {
                colors[i][j] = Color.WHITE;
            }

        }
    }
}
```

Alan's Pattern





```
public void makePattern()
{
    for (int i = 0; i < ROWS; i++)
    {
        for (int j = 0; j < COLUMNS; j++)
        {
            if ((i + j) % 3 == 0)
            {
                colors[i][j] = Color.BLUE;
            }
            else if ((i + j) % 3 == 1)
            {
                colors[i][j] = Color.RED;
            }
            else if ((i + j) % 3 == 2)
            {
                colors[i][j] = Color.WHITE;
            }

        }
    }
}

// Don't look at the code below
```

Scribe: Froilan Zarate
Driver: Alan Xiao

The Lab Assignment:

You will create a class SequenceAlignment.
In that class, you will create a single method called alignSequences(String, String).
It will return an int, containing the score of the alignment.

Creating the Grid:

Follow the example to see how you should set up the program.

Let's say that we were using president names instead of ACGT's from genetics.
HOOVER
ROOSEVELT

You will create a 2d int array with the size of
Length = word1+1
Height = word2+1
Initialize the top left square as a 0. This is to represent the fact that, for two empty strings (prefixes of length 0 of HOOVER and ROOSEVELT), we will say that they have 0 similarity score. The score for each square will be calculated starting from here. Each square in the first row will be equal to the number on its left minus two. This is because moving horizontally across the grid is considered a gap: the -8 under the V represents that we have matched the length 4 prefix HOOV with 4 gaps preceding the R in ROOSEVELT, at a cost of 8.
The same occurs for each element in the first column.
A visualization has been shown below. Note that the very first white column and row do not actually exist within the grid, they are just there to help keep track of the prefix lengths and indices within the grid, along with the letter from the string that each index represents. You need to keep track of what column and row corresponds to which letter. Note that the grid indices for the problem are off by 1 from what the characters would be indexed within the string: H in "HOOVER" is at index 0, but here we reserve column 0 in the grid to hold matching the empty string prefix of HOOVER against different prefixes of ROOSEVELT, while prefix "H" has length 1.