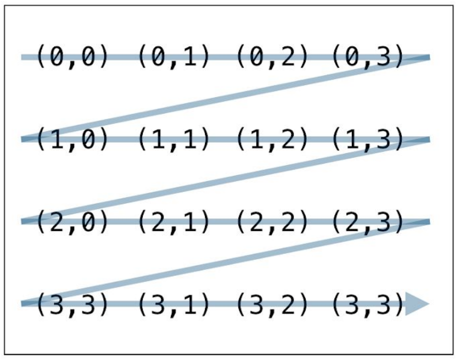
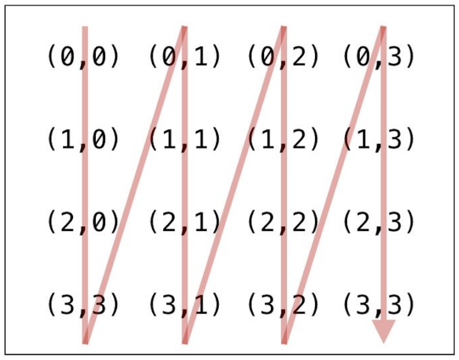
EXERCISE 1: A Grid Iterator

Download **`Grid.zip`** from the precepts page, unzip the project and open it using IntelliJ.

(a) Implement the **`GridIterator`** class in **`Grid.java`** to enable iterating over the elements in the grid in row-major order (as shown below). Test your program by running the given driver program.



Row−major



Column−major

(b) Create another iterator **`ColMajorIterator`** that returns elements in *column-major* order. Add code to **`main`** that prints the grid elements using this iterator.

(c) Convert **`Grid.java`** to an *Iterable*, where the default iteration is in row-major order. Test your code by converting the while loop in **`main`** to a for-each loop.

(d) Consider the following piece of code, where **`myGrid`** is an object of type **`Grid<Integer>`**:

```java
boolean flag = true;
for (int i : myGrid) {
    int count = 0;
    for (int j : myGrid)
        if (i == j)
            count++;
    if (count > 1) {
        flag = false;
        break;
    }
}
StdOut.println(flag);
```

- What does this code do?
- What is the order of growth of the running time of this code?
- How many iterator objects does this code create?

EXERCISE 2: Memory Analysis

(a) How much memory does each of the following pieces of code use as a function of the input size *n*? Use tilde notation to simplify your answer.

(***Note***: An object of type **`Double`** uses 24 bytes of memory, whereas a **`double`** variable uses 8 bytes only)

(1)

```
double[] a = new double[n];
```

(2)

```
double[] a = new double[n];
for (int i = 0; i < n; i++)
    a[i] = 0.5;
```

(3)

```
Double[1][] a = new Double[n];
```

(4)

```
Double[] a = new Double[n];
for (int i = 0; i < n; i++)
    a[i] = new Double(0.5);
```

[1] This example should not be interpreted to mean that creating an array of type Double[] instead of double[] is a good idea. It is actually a bad idea! Do not use the wrapper type unless you are forced to (like in generics).

(b) Use tilde notation to describe how much memory an object of type **`Grid<Item>`** requires as a function of ***n*** right after the constructor finishes execution. Note that the grid is of size ***n*** x ***n***.

(c) Use tilde notation to describe how much memory a **`Grid<Integer>`** object requires as a function of ***n***, assuming that there are no *null* items in the grid. Note that every object of type **`Integer`** requires 24 bytes.

(d) Use tilde notation to describe how much memory a **`GridIterator<Integer>`** object requires as a function of ***n***.