1 Packages have arrived

In the following classes, cross out the lines that will result in an error (either during compilation or execution). Next to each crossed-out line write a replacement for the line that correctly carries out the evident intent of the erroneous line.

Each replacement must be a single statement. Change as few lines as possible.

After your corrections, what is printed from running `java P2.C5`?
2 Max Pooling

In this question, you will implement a function for performing max pooling, which is a technique used in deep learning for reducing the resolution of an image being fed through the layers of a neural network. (Summer 2019, MT1)

Images can be represented by two-dimensional arrays; the first dimension represents the rows of the image, and the second dimension is to represent the value of the pixel at a particular column index within a row.

For example, given a 2-dimensional image int[][] array called img, the pixel in the 5th row, and the 7th column can be accessed via img[5][7].
One way we can implement max pooling is by cutting up our image \( \text{img} \) into small equal sized squares, and taking only the largest pixel value in each piece as the representative for that piece in our final downsampled image.

For example, this 4 by 4 \( \text{img} \) can be broken down into 4 equal-sized pieces of shape \( \text{windowSize} \) by \( \text{windowSize} \) (where \( \text{windowSize} \) is 2). The largest value in each of these pieces is recorded in result in their corresponding locations.

```
static int[][] maxPool(int[][] img, int windowSize) {
    // resRows are the number of rows in result.
    // resCols are the number of columns in result.
    int resRows = _____________________________________;
    int resCols = _____________________________________;
    int[][] result = __________________________________;
    for (int r = _______; ___________ ; ________) {
        for (int c = _______; ___________ ; ________) {
            // Java's Math.max() function only accepts two arguments at a time.
            // (Put one on the first line, and the second on the line below it).
            int largestSoFar = Math.max(______________________________,
                                          ____________________________________________);
            ____________________________________________ = largestSoFar;
        }
    }
    return result;
}
```
3 Iterator of Iterators

Implement an `IteratorOfIterators` which will accept as an argument a `List` of `Iterator` objects containing `Integers`. The first call to `next()` should return the first item from the first iterator in the list. The second call to `next()` should return the first item from the second iterator in the list. If the list contained `n` iterators, the `n+1`th time that we call `next()`, we would return the second item of the first iterator in the list.

For example, if we had 3 `Iterators` A, B, and C such that A contained the values [1, 2, 3], B contained the values [4, 5, 6], and C contained the values [7, 8, 9], calls to `next()` for our `IteratorOfIterators` would return [1, 4, 7, 2, 5, 8, 3, 6, 9].

Feel free to modify the input `a` as needed.

Note - this is only one possible solution, as there are many others.

```java
import java.util.*;

public class IteratorOfIterators implements _______________ {
    LinkedList<Integer> l;

    public IteratorOfIterators (ArrayList<Iterator<Integer>> a) {
        l = ________________;
        int i = 0;

        while (______________) {
            Iterator<Integer> curr = ________________;
            if (!curr.hasNext()) {
                ________________;
                ________________;
            } else {
                ________________;
            }
            if (a.isEmpty()) {
                ________________;
            }
            i = ________________;
        }

        @Override
        public boolean hasNext() {
            return ________________;
        }

        @Override
        public Integer next() {
            return ________________;
        }
    }
}
```