A for-loop is sometimes called a "definite loop" because for-loops are often used when it is known how many times to repeat some lines of code. For example the following for-loop executes the code inside the curly-braces 5 times.

```
for (int i=0; i<5; i++) {

}
```

A for loop with the pattern

```
for (<init>; <test>; <update>) {
  <code>
}
```

Is nearly equivalent to pseudocode with this pattern

```
<init>
while <test> is true
  <code>
  <update>
```

1) What sequence of values does `i` take on in the above example?

The `i++` part of a for-loop's control can be more complicated. For example, if you wanted `i` to take on only the even values less than 5, `for (int i=0; i<5; i=i+2)` would do it. The `i++` was replaced with `i=i+2`, which is used to update `i` after each iteration.

2) What sequence of values does `i` take on for each of the following for-loops?

```
for (int i=0; i<5; i=i+2)

for (int i=1; i<=5; i++)

for (int i=1; i<=5; i=i+2)

for (int i=0; i<5; i=i+2)

for (int i=64; i!=0; i=i/2)
```

3) Write a for-loop control that makes `i` take on the following sequences.

1, 2, 3, 4

4, 3, 2, 1

1, 2, 4, 8, 16

9, 7, 5, 3, 1

Nested for-loops look like this and can be tricky to write

```
for (int i=0; i<5; i++) {
    for (int j=0; j<3; j++) {

    }
}
```

The outer for-loop causes the inner for-loop to be run 5 times. Each time the inner for-loop is run, it executes the code in the inner curly-braces 3 times. So in this example, the inner curly-brace block is run a total of 15 times.

PracticeIt / CodeStepByStep has multiple practice problems where you are asked to draw figures using nested for loops. For example

```
*****
*****
*****
```

You can see that this is 3 lines, each with 5 asterisks. The big picture in pseudocode can be expressed

```
for line = 1, 2, 3
    draw line with 5 asterisks
```

And draw line with 5 asterisks can be written in pseudocode as

```
for i = 1, 2, 3, 4, 5
    print *
newline
```

If you combine the two you get

```
for line = 1, 2, 3
    for i = 1, 2, 3, 4, 5
        print *
    newline
```

4) Rewrite this pseudocode in Java.

5) Follow the same procedure to write nested for-loops for the following figure: (i) write pseudocode for the outer for-loop, (ii) write pseudocode for the inner for-loop, (iii) combine the pseudocodes, and (iv) covert the pseudocode to Java.

```
*
**
***
****
*****
```