

NAME _____

DATE _____

PERIOD _____

UNIT 2 · TOPIC 2.12 · WORKSHEET

Count Every Step

Read a loop header and know how many times it runs, count the work inside nested loops, and catch the loop that runs one time too many.

About 35 minutes

10 questions

Cumulative through 2.12

TOPIC

2.12

**DIRECTIONS**

Work each puzzle in pencil. A statement's **execution count** is how many times it runs while the program runs. For code, write exactly what prints. Some puzzles include pieces you won't need.

PART A · 2 QUESTIONS

A

Warm up your brain

1

YOU'RE THE JVM

Run this code in your head. The loop counts down by 7. What prints?

```
int t = 30;
int ticks = 0;
while (t > 5) {
    t -= 7;
    ticks++;
}
System.out.println(ticks + " " + t);
```

A 4 2**B** 3 9**C** 5 -5**D** 5 2

2 PENCIL CHECK

Read each loop header. How many times does the loop's body run? List the values of k if it helps.

Loop header	Body runs
<code>for (int k = 0; k < 12; k++)</code>	
<code>for (int k = 1; k <= 12; k++)</code>	
<code>for (int k = 4; k <= 12; k++)</code>	
<code>for (int k = 0; k < 12; k += 3)</code>	
<code>for (int k = 12; k > 0; k -= 5)</code>	
<code>for (int k = 7; k < 3; k++)</code>	



THINK FIRST

A nested loop does some work for an input of size 10. If the input doubles to size 20, does it do twice as much work? Guess in the margin; question 9 settles it.



WATCH OUT

Count with the braces, not the indentation. A loop with no braces repeats only the **one** statement right after its header, however the next lines are indented. Find the `{` and `}` (or that single statement) before you count.

PART B · 4 PUZZLES

B Puzzles

3 CODE TILES

This program shows that a loop's condition is checked one more time than its body runs: the last line prints the value that finally made the condition false. The first line stayed on the fridge. Put the tiles back in order under it so the program prints the target output. One tile is a decoy and isn't used.

```
int lap = 1;
```

lap++;	while (lap < 3) {
System.out.println("now " + lap);	}
System.out.println("lap " + lap);	while (lap <= 3) {

Target output

lap 1

lap 2

lap 3

now 4

YOUR PROGRAM

4 SNIPPET POOL

Fill each numbered hole with a loop header part from the pool so the program prints 8 5 12. Each snippet is used at most once.

Counts.java

```
int a = 0;
int b = 0;
int c = 0;
for ( (1) ) {
    a++;
}
for ( (2) ) {
    b++;
}
for ( (3) ) {
    for (int j = 0; j < 3; j++) {
        c++;
    }
}
System.out.println(a + " " + b + " " + c);
```

THE POOL

Each snippet can be used only once, and not every snippet is needed.

int k = 1; k < 8; k++	int k = 0; k < 10; k += 2
int k = 0; k <= 8; k++	int k = 1; k <= 8; k++
int k = 0; k <= 10; k += 2	int k = 20; k > 0; k -= 5

5 MATCH-UP

Each segment counts its work in `x`, which starts at 0. When `n` is 10, how many times does `x++` run? Match each segment to its count from the list below. Each count is used once.

Segment	Code	<code>x++</code> runs
1	<pre>for (int k = 0; k < n; k++) { x++; }</pre>	
2	<pre>for (int i = 0; i < n; i++) { for (int j = 0; j < n; j++) { x++; } }</pre>	
3	<pre>int size = n; while (size > 1) { size /= 2; x++; }</pre>	
4	<pre>for (int i = 0; i < n; i++) { for (int j = 0; j < i; j++) { x++; } }</pre>	

- **P** 3
- **Q** 10
- **R** 45
- **S** 100

6 FACE-OFF

Tonight: a loop's Condition and its Body argue about who works harder. Fill in the blanks.

CONDITION I get checked before every pass, plus _____ more time at the end. That last check is the one that comes out false.

BODY I run once per pass. Ten passes, ten of me.

CONDITION And for those ten passes, I'm checked _____ times. I win.

BODY Not always. If you're false the very first time, I run _____ times, and you're checked only _____.

CONDITION Fine. Then picture while (size > 1) with size /= 2; in your body, and size starting at 64.

BODY Easy. 64, 32, 16, 8, 4, 2: I run _____ times, and you're checked 7.



FAIR QUESTION

Why count statements? Why not just time the program?

A stopwatch measures your computer, not your code. The same program runs faster on a new laptop than on an old one, and even changes from run to run. An execution count is the same everywhere, so it's a fair way to compare two pieces of code.

PART C · 4 QUESTIONS

C

Catch the mistake

7

WHICH DOES LESS WORK?

Both segments add up the multiples of 5 from 5 to 100, and both end with sum equal to 1050.

Segment A

```
int sum = 0;
for (int k = 1; k <= 100; k++) {
    if (k % 5 == 0) {
        sum += k;
    }
}
```

Segment B

```
int sum = 0;
for (int k = 5; k <= 100; k += 5) {
    sum += k;
}
```

How many times does each loop's **body** run?

A A: 100, B: 20**B** A: 20, B: 20**C** A: 100, B: 21**D** A: 101, B: 20

How many times does `sum += k;` run in each segment? Which segment does less work?

YOUR ANSWER

8 ONE-TOKEN FIX

This loop is supposed to beep exactly 10 times, but its body runs 11 times and it prints 11. There are **three** different one-token changes that make the body run exactly 10 times. Find all three.

```
1 int beeps = 0;
2 for (int k = 0; k <= 10; k++) {
3     beeps++;
4 }
5 System.out.println(beeps);
```

THREE FIXES (OLD TOKEN TO NEW TOKEN)

9 TWO TRUTHS AND A LIE

This nested loop counts its own work. Two of the claims are true and one is a lie. Circle the lie, then rewrite it so it's true.

```
int work = 0;
for (int i = 0; i < n; i++) {
    for (int j = 0; j < n; j++) {
        work++;
    }
}
System.out.println(work);
```

- **1** When n is 10, `work++` runs 100 times.
- **2** When n doubles to 20, `work++` runs 200 times, twice as many.
- **3** The `println` runs exactly once, whatever n is.

THE LIE, REWRITTEN

10 WRAP-UP**REFLECTION**

In one sentence: why can three lines of code do more work than thirty?

YOUR ANSWER