

NAME

DATE

PERIOD

UNIT 2 · TOPIC 2.1 · WORKSHEET

Decide and Repeat

Sort the steps, trace the frog, match the loops, and catch the flowchart that's fibbing.

About 35 minutes

10 questions

Cumulative through 2.1

TOPIC

2.1



DIRECTIONS

Work each puzzle in pencil. For Java code, write exactly what prints. In the written algorithms, a "Repeat until" step checks its goal before every pass and stops as soon as the goal is true. Some puzzles include pieces you won't need.

PART A · 2 QUESTIONS

A

Warm up your brain

1

PENCIL CHECK

Each step comes from an everyday algorithm. Write the building block it shows: *sequencing*, *selection*, or *repetition*.

Step	Building block
Preheat the oven to 350 degrees.	
Knead the dough until it is smooth.	
If the light is green, cross the street.	
Write your name at the top of the test.	
Keep pressing snooze until it is 7:00.	
If the bread is golden, take it out; otherwise, wait two more minutes.	

2 TRACE THE FROG

Hopper the frog follows this algorithm. Trace it in your head. What does Hopper say, and where does Hopper end up?

```
1 Set position to 0 and hops to 0.
2 Repeat until position is at least 10:
3     add 4 to position, then add 1 to hops.
4 If hops is more than 2, say "long trip";
5     otherwise, say "short trip".
```

- A** It says "short trip" and ends at 12.
- B** It says "long trip" and ends at 12.
- C** It says "long trip" and ends at 10.
- D** It says "short trip" and ends at 8.

**THINK FIRST**

A game adds a 5-point bonus to your score and then doubles it. Would you rather it doubled first and added the bonus after? Guess in the margin; question 3 settles it.

PART B · 3 PUZZLES

B Puzzles

3 CODE TILES

Hopper's game gives a 5-point bonus at the end of a level, then doubles the score. Someone knocked the program off the fridge. Put the tiles back in order so it prints the output below. One tile is a decoy and isn't used.

<code>System.out.println("final: " + score);</code>	<code>score *= 2;</code>
<code>int score = 40;</code>	<code>System.out.println("bonus: " + score);</code>
<code>score += 5;</code>	<code>score += 10;</code>

Target output

bonus: 45

final: 90

YOUR PROGRAM



FAIR QUESTION

That program had no decision and no loop. Why is it here?

Because order matters for every building block, not just sequencing. When an algorithm combines sequencing, selection, and repetition, moving a step inside a loop, out of a loop, or before a decision can change what it produces, the same way swapping two lines did.

4 MATCH-UP

Four loops look almost the same. Each one changes one thing. Match each loop to what it says, from the list below. Each output is used once.

Loop	Algorithm	Output
1	Set count to 0. Repeat until count is 3: add 1 to count say count	
2	Set count to 0. Repeat until count is 3: say count add 1 to count	
3	Set count to 1. Repeat until count is 3: say count add 1 to count	
4	Set count to 0. Repeat until count is 3: add 1 to count Say count.	

- **P** 3
- **Q** 0 1 2
- **R** 1 2
- **S** 1 2 3



WATCH OUT

"Knock. Knock. Knock." is three sequencing steps, not repetition. It also guesses that three knocks is enough. "Knock until someone answers" is repetition: one instruction, and the goal decides when to stop.

5 FACE-OFF

Tonight: Selection and Repetition argue about who does more work. Fill in the blanks.

SELECTION	I ask a question whose answer is _____ or _____, and the answer chooses which step happens next.
REPETITION	Cute. You ask your question _____ and you're done. I keep going until a _____ is reached.
SELECTION	Without me, "take an umbrella" would happen _____ day, rain or shine.
REPETITION	Without me, you'd have to guess how many times to stir the batter and write "stir" that many times.
SELECTION	At least my questions always end. If your goal can never be reached, you run _____.
REPETITION	That's called an _____ loop, and it's the algorithm's fault, not mine.

PART C · 5 QUESTIONS

C Catch the mistake

6 ONE-NUMBER FIX

This algorithm never stops. Change exactly one number on line 3 so that it stops after exactly 3 passes through the loop.

```

1 Set x to 1.
2 Repeat until x is 10:
3     add 2 to x.
4 Say x.
```

WHY IT NEVER STOPS, AND YOUR FIX

7 TWO TRUTHS AND A LIE

This flowchart sets a movie ticket price. Two of the claims about it are true and one is a lie. Circle the lie, then rewrite it so it's true.

```

(Start)
  |
[Read age]
  |
< Is age less than 12? > --true--> [Price is 5] --+
  | false                                     |
[Price is 9]                                     |
  | <-----+
[Say price]
  |
(Stop)

```

- 1 For age 8, it says 5.
- 2 For age 12, it says 9.
- 3 Every box runs once each time, so the price always ends up as 9.

THE LIE, REWRITTEN

8 YOU'RE THE JVM

You are the Java Virtual Machine. Run this code in your head. What prints?

```

String pond = "lilypad";
System.out.println(pond.indexOf("p") + " " + Math.abs(3 - 8));

```

- | | |
|---------------|---------------|
| A 4 5 | B 5 5 |
| C 4 -5 | D 5 -5 |

There's no decision or loop in this code, but each method's result comes from one. Which building block does `indexOf` need, and which does `Math.abs` need?

YOUR ANSWER

9 WHO AM I?

Some algorithm ideas are playing a party game. Each gives a clue about itself. Name the idea.

Clue	I am...
I'm the backbone: steps one after another, each in its place.	
I ask a true-or-false question and pick the path.	
I keep doing the same steps until a goal is reached.	
In a flowchart, I'm a diamond with a true arrow and a false arrow.	
I'm a loop whose goal can never be reached, so I never stop.	

10 WRAP-UP



REFLECTION

In one sentence: why can't a pure sequence of steps handle a rainy day and a sunny day differently?

YOUR ANSWER