Where's the popcorn?

An introduction to while loops in Java.

While loops are loops that continue to loop while a condition is true, here's what one looks like:

```java
while (condition) {
	// code that loops
}
```

For example, if you were programming a car to stay on the highway until exit 11:

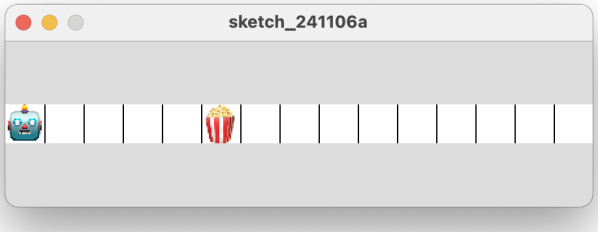
```java
while (nextExit != 11) {
	driveForward();
}

exitHighway();
```

In this worksheet, we're going to be using while loops to program a robot to find some popcorn.

Problem 1

Get started by downloading the starter code. Open it in Processing, click *Run*, and you should see this screen:



Your goal is to program the robot to move right towards the popcorn until it's touching the popcorn.

You can control the robot with the following commands:

```java
robot.moveLeft(); // moves the robot one square to the left
robot.moveRight(); // moves the robot one square to the right
```

And to check if the robot is touching the popcorn:

```java
robot.isTouching(popcorn); // returns true when touching, otherwise, false
```

Write you code inside of the *problem1()* method and when done copy it here:

```java
void problem1() {
  // here's where your code goes for problem 1!
}
```

When it works, you'll see a green screen! Make sure it works consistently.

Hint: this problem could be rephrased as: program the robot to move right towards the popcorn while it is not touching the popcorn.

Problem 2

Congrats on finishing problem 1! To switch problem, change line 1 to:

```java
int problem = 2;
```

Now, try running the program. You should see a slightly different situation. In this problem, the popcorn will either be to the right or the left of the robot.

Copy your code from the *problem1()* method into the *problem2()* method. How can you modify this code to handle this slightly more difficult situation? Here are two new methods you might find helpful:

```java
robot.canMoveLeft(); // returns true or false
robot.canMoveRight(); // returns true or false
```

Once you're done, copy your working code below:

```java
void problem2() {
  // here's where your code goes for problem 2!
}
```

Remember, you also have the following methods:

```java
robot.moveLeft(); // moves the robot one square to the left
robot.moveRight(); // moves the robot one square to the right
robot.isTouching(popcorn); // returns true when touching, otherwise, false
```

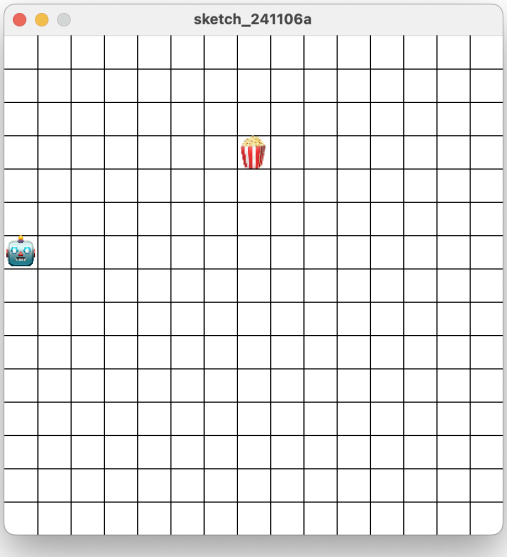
Remember, a green screen indicates success.

Problem 3

Let’s get started by switching to problem three:

```java
int problem = 3;
```

And wow! When you run that code, you’ll see something very different:



It’s the same puzzle as before, however, now it’s on a 2D plane. To help you tackle this problem, here are some new methods available:

```java
robot.moveUp(); // moves the robot one square upwards
robot.canMoveUp(); // returns true or false
robot.moveDown(); // moves the robot one square done
robot.canMoveDown(); // returns true or false
robot.isOnTheSameXAxisAs(popcorn); // returns true when on the same x-axis
robot.isOnTheSameYAxisAs(popcorn); // returns true when on the same y-axis
```

It might take your robot a little bit of time to reach the popcorn, that’s alright! Slow and steady wins the race, right?

One helpful way to think about this problem is to first get your robot on the same x-axis and then work on getting them to touch.

When you’re done, copy your code here:

```java
void problem3() {
  // here's where your code goes for problem 3!
}
```

Problem 4

Let’s switch over to problem four:

```java
int problem = 4;
```

And you’ll notice that it’s essentially the same problem! There’s one catch, however, now you need to eat the popcorn once you’re touching it.

Here’s the command to eat popcorn:

```java
robot.eat(popcorn);
```

Now, you can only do this when you’re touching the popcorn.

When you’re done, copy your code below:

```java
void problem4() {
  // here's where your code goes for problem 4!
}
```

Challenge: Problem 5

Let's switch over to problem five:

```java
int problem = 5;
```

And now there are multiple boxes of popcorn! For this challenge, you're going to need to eat all of the popcorn. Here are some things to know:

1) *count* is a variable whose value is the amount of popcorn on the screen when the game starts.
2) *popcorn* is no longer defined because there are multiple, instead you'll need to use the "scanner" to find the nearest popcorn:

```java
Popcorn nearest = scanner.nearestPopcorn();
```

Here are some questions to think about?

1) How can you break this into multiple versions of Problem 4?
2) How can you keep track of how many boxes of popcorn you've eaten?

When you're finished, copy your code below:

```java
void problem5() {
  // here's where your code goes for problem 5!
}
```