Matching AgentSheets Code to the underlying Java code
Can you read this code?
How about this code?

```c
else if (see(0, -1, "truck")) {
    play_sound("honk");
    change(0, 0, "squished_frog");
    wait("0.5");
    erase(0, 0);
    reset_simulation();
}
```
What does this code say?

```java
public void while_running() {
    if (stacked_a("immediately above", "a", "tunnel")) {
        say("cannot walk over tunnels. that's cheating!");
        erase(0, 0);
        stop_simulation();
    }
}
```
Here’s another... which agent was this written for?

```java
public void while_running() {
    if (see(0, 1, "tunnel_right")) {
        erase(0, 0);
    } else if (see(0, 1, "street") && once_every("0.5", "secs")) {
        move(0, 1);
    }
}
```
Now... this is the code for the ‘large fish’ from a totally different simulation. Can you figure out what’s going on?

```java
public void while_running() {
    if (is("hunger", ">", "100")) {
        erase(0, 0);
    }
    else if (see(0, -1, "fish_small") && %_chance("15")) {
        erase(0, -1);
        set("hunger", "to", "0");
    }
    else if (see(1, 0, "water") && %_chance("1")) {
        new(1, 0, "fish_large");
        set("hunger", "to", "hunger + 2");
    }
    else if (true) {
        move_random_on("water");
        set("hunger", "to", "hunger + 1");
    }
}
```
If this code snippet runs, what happens to @small? (Note, one is the ‘while running’ code, and the second is the method ‘count’)

```java
public void while_running() {
    if (true) {
        set("@small", "to", "0");
        set("@large", "to", "0");
        broadcast("fish_small", "count");
        broadcast("fish_large", "count");
    }
}

public void count() {
    if (true) {
        set("@small", "to", "@small + 1");
    }
```
7. The question below uses a simple programming language, with the following instructions.

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>random(a, b)</td>
<td>Generates a random integer between a and b, inclusive.</td>
</tr>
<tr>
<td>a + b</td>
<td>Evaluates to the sum of the numbers a and b.</td>
</tr>
<tr>
<td>a &lt;- b</td>
<td>Assigns the value of b to the variable a.</td>
</tr>
<tr>
<td>REPEAT n TIMES { }</td>
<td>The block of instructions contained between the braces { } is repeated n times.</td>
</tr>
<tr>
<td>display(expression)</td>
<td>Displays the value of expression.</td>
</tr>
</tbody>
</table>

Consider the goal of simulating the results of rolling a number cube (numbered 1 to 6) two times, and displaying the sum of the values obtained from the two rolls. Which of the following code segments will produce the appropriate results?

I. `display(random(1, 12))`
II. `display(random(1, 6) + random(1, 6))`
III. `sum <- 0
     REPEAT 2 TIMES
     {
       sum <- sum + random(1, 6)
     }
     display(sum)`

(A) I only
(B) I and III only
(C) II and III only
(D) I, II, and III