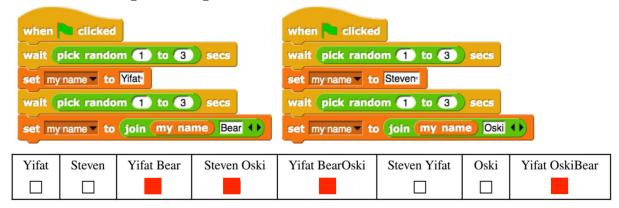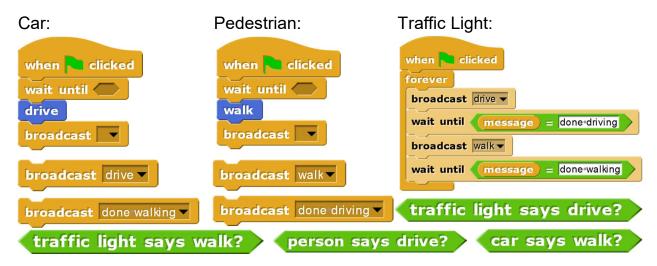# Concurrency and Paradigms

Con-What-Curr-Is-En-My-Cy-Name

(a) **Which of the following could be the value of the variable `my_name` when the green flag is clicked?**

```
when [flag] clicked
wait (pick random (1) to (3)) secs
set [my name v] to [Yifat]
wait (pick random (1) to (3)) secs
set [my name v] to (join (my name) [Bear])
```

```
when [flag] clicked
wait (pick random (1) to (3)) secs
set [my name v] to [Steven]
wait (pick random (1) to (3)) secs
set [my name v] to (join (my name) [Oski])
```

Yifat	Steven	Yifat Bear	Steven Oski	Yifat BearOski	Steven Yifat	Oski	Yifat OskiBear
☐	☐	🟥	🟥	🟥	☐	☐	🟥

For the next two problems, suppose we have these three scripts and eight blocks:

Car:

```
when [flag] clicked
wait until < >
drive
broadcast [ v]
```
```
broadcast [drive v]
```
```
broadcast [done walking v]
```
```
<traffic light says walk?>
```

Pedestrian:

```
when [flag] clicked
wait until < >
walk
broadcast [ v]
```
```
broadcast [walk v]
```
```
broadcast [done driving v]
```
```
<person says drive?>
```

Traffic Light:

```
when [flag] clicked
forever
  broadcast [drive v]
  wait until <(message) = [done driving]>
  broadcast [walk v]
  wait until <(message) = [done walking]>
```
```
<traffic light says drive?>
```
```
<car says walk?>
```

(b) **Fill in the code for the car and the pedestrian such that there will be a deadlock where the pedestrian and the car rely on each other to know when to drive or walk, respectively. You may not need to fill all the cells.**

	car	pedestrian
`waits until…`	<person says drive?>	<car says walk?>
`broadcasts…`		

(c) Now fill in the code for the car and the pedestrian so that they function properly, relying only upon the traffic light. You may not need to fill in all the cells.

	car	pedestrian
`waits until…`	<traffic light says drive?>	<traffic light says walk?>
`broadcasts…`	[broadcast (done driving)]	[broadcast (done walking)]

Paradigm City

(a) Match each of the following to one of the four programming paradigms (functional, OOP, declarative, imperative).

One sprite tells a second sprite to run some code. The second sprite does it.

OOP

You input a board into a `copy_of_board` block. The block outputs a new board without changing the original board.

functional

You give a program a condition as an input and it uses this condition to remove numbers from a list. You input a list and it removes items.

declarative

You have a global variable set to a secret word. You change the secret word every time you ask a player for a new secret word.

imperative

(b) Match each of the following scripts to a programming paradigm.

functional

imperative

OOP

Assume sum *is a global variable in the second script*