

## Discussion 5: Concurrency

### Concurrency

1. CS10 has decided to open a pizzeria! To make a pizza, the following tasks must be completed:

Task	Time
Make the dough	25 minutes
Make the sauce	25 minutes
Prepare the toppings	10 minutes
Assemble the pizza	10 minutes
Bake the pizza	50 minutes

a. Which of these tasks must be completed in serial?

Assemble Pizza, Bake pizza

---

b. Which of these tasks can be completed in parallel?

Make dough, make sauce, prepare toppings

---

c. Based on Amdahl's Law, how fast can we make a single pizza? 1 hour  
(in real life 1 hr 25 minutes)

d. How many employees would the pizzeria need to make a pizza this fast? infinity  
(in real life 3 workers)

e. Would adding an employee to your answer from part (d) change the time it takes to make a pizza?

No it would not because we are limited by the serial portion

---

2. Assume we click the green flag to run the code below, then wait 60 seconds. What are all the possible values of magic after 60 seconds have elapsed?

```

when clicked
  set magic to X
  broadcast Magic Show
  
```

```

when I receive Magic Show
  wait pick random 1 to 5 secs
  repeat until magic = A or magic = B
    broadcast Magic Show
  set magic to C
  
```

```

when I receive Magic Show
  wait pick random 1 to 5 secs
  repeat until magic = B or magic = C
    broadcast Magic Show
  set magic to A
  
```

```

when I receive Magic Show
  wait pick random 1 to 5 secs
  repeat until magic = A or magic = C
    broadcast Magic Show
  set magic to B
  
```

Possible values of magic: X the code will get caught in a deadlock

3. Which of the following could be the value of my\_name after the green flag is clicked?

```

when clicked
  wait pick random 1 to 3 secs
  set my name to Dan
  wait pick random 1 to 3 secs
  set my name to join my name Oski
  
```

```

when clicked
  wait pick random 1 to 3 secs
  set my name to Garcia
  wait pick random 1 to 3 secs
  set my name to join my name Bear
  
```

- Dan
- Garcia
- Dan Bear
- Garcia Oski
- Dan BearOski
- Garcia Dan
- Oski
- Dan OskiBear

### Testing

The following questions are based off this block:

```

+square+numbers+in+ list +
script variables new
set new to list
for i = 1 to length of new
  replace item i of new with item i of new x item i of new
report new
  
```

1. We try to test our code, but we get an error. What does it mean and how can we fix it?

```

test square numbers in w/ inputs list 1 2 expecting output
list 1 4
  
```

Inside: Error expecting list but getting number

The test block requires that all inputs for the function being tested be contained in a list. Here the input is a two-item list, so this list must be inserted into an outer list. The correct input should be list(list(1, 2)). Note that this is not true for the outputs (i.e., the output given here is formatted correctly). As a side note, if you ever see the error message above, you are probably using lists incorrectly somewhere in your code, likely by passing an invalid argument to a function.

2. Now, we try to run the following test, but it doesn't work as expected:

```

set my list to list 1 2
test square numbers in w/ inputs
report list my list list my list expecting output
list 1 4 list 1 4
  
```

1  true

2  false

length: 2

Why does it output this, and how could we fix it?

square numbers is mutating the input list! The first time we run square numbers with my list, the list is changed to [1, 4]. That means that the second time we try to run square numbers with input my list, we are running the function with input [1, 4]. This gives us output [1, 16] (not matching our expectations, and thus failing the test). We can fix this by making sure that square numbers is not using the replace block to mutate the input list. Instead, it should square the values in the input list and add them to a new output list (perhaps by using HOFs).

- Assuming we haven't changed the code for square numbers, what should we expect this block to output? Is it any different from the output from part 2?

```

test square numbers in [ ] w/ inputs
list list 1 2 <<>> list list 1 2 <<>> expecting output
list 1 4 <<>> list 1 4 <<>>

```

The test block will output [True, True] here. square numbers still mutates the input list, but because we are using a different input list each time we call square numbers, the mutation is not propagated between test cases. This example highlights the (often tricky) subtlety of mutability, as well as the necessity of having comprehensive test cases. If our test cases did not try using the same list as input twice, we would not catch the bug.

### Challenge

- List all possible values of grade after the green flag is clicked.

```

when green flag clicked
  set Grade to 10
  broadcast Apply Final Grading

when I receive Apply Final Grading
  Apply EPA (Get Grade - 5)

when I receive Apply Final Grading
  set Grade to (Get Grade * Get Grade)

```

Here are the definitions of the blocks used in the above scripts:

```

+ Apply + EPA + points +
wait pick random 1 to 10 secs
change Grade by points

+ Get + Grade +
wait pick random 1 to 5 secs
report Grade

```

Possible values of grade :

Using a strategy similar to those described above, we can compute that the possible values are 225, 150, 105, and 195.