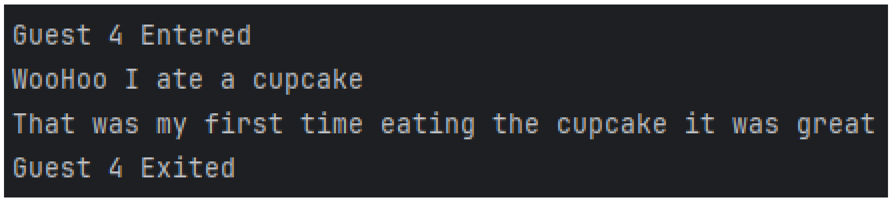
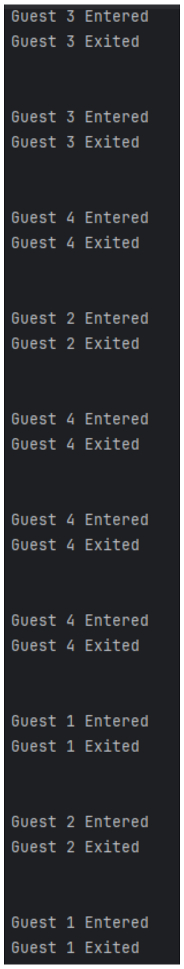
Part 1

Statement of correctness and strategy:

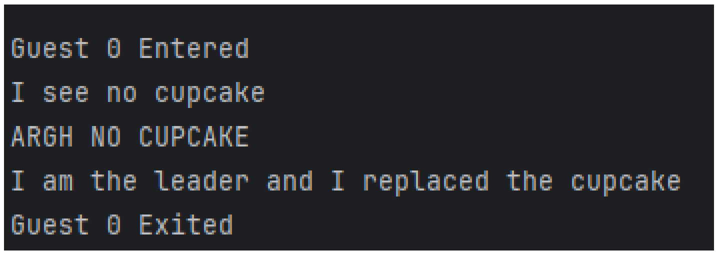
Set one guest as the leader who is the only one to replace the cupcake and count how many people ate the cupcake. Everyone only eats the cupcake once if they see the cupcake again, they pass by it if they have already had one. This means the cupcake only disappears when someone sees it for the first time. That's how the leader knows that someone new ate a cupcake. Once he counts as many replacements as there are players everyone can say that they have eaten the cupcake and won the game. When tested on multiple different inputs, all guest eat the cupcake without communicating with each other. The use of a semaphore only allows one guest in at a time.



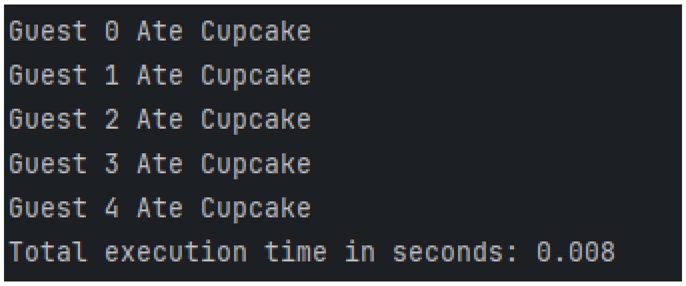
```
Guest 4 Entered
WooHoo I ate a cupcake
That was my first time eating the cupcake it was great
Guest 4 Exited
```



```
Guest 3 Entered
Guest 3 Exited


Guest 3 Entered
Guest 3 Exited


Guest 4 Entered
Guest 4 Exited


Guest 2 Entered
Guest 2 Exited


Guest 4 Entered
Guest 4 Exited


Guest 4 Entered
Guest 4 Exited


Guest 4 Entered
Guest 4 Exited


Guest 1 Entered
Guest 1 Exited


Guest 2 Entered
Guest 2 Exited


Guest 1 Entered
Guest 1 Exited
```



```
Guest 0 Entered
I see no cupcake
ARGH NO CUPCAKE
I am the leader and I replaced the cupcake
Guest 0 Exited
```



```
Guest 0 Ate Cupcake
Guest 1 Ate Cupcake
Guest 2 Ate Cupcake
Guest 3 Ate Cupcake
Guest 4 Ate Cupcake
Total execution time in seconds: 0.008
```

Efficiency:

For 5 inputs it take 8 ms, due to the use of a semaphore only allowing one thread into the labyrinth at a time this is likely the fastest possible time

Part 2:

Strategy 1:

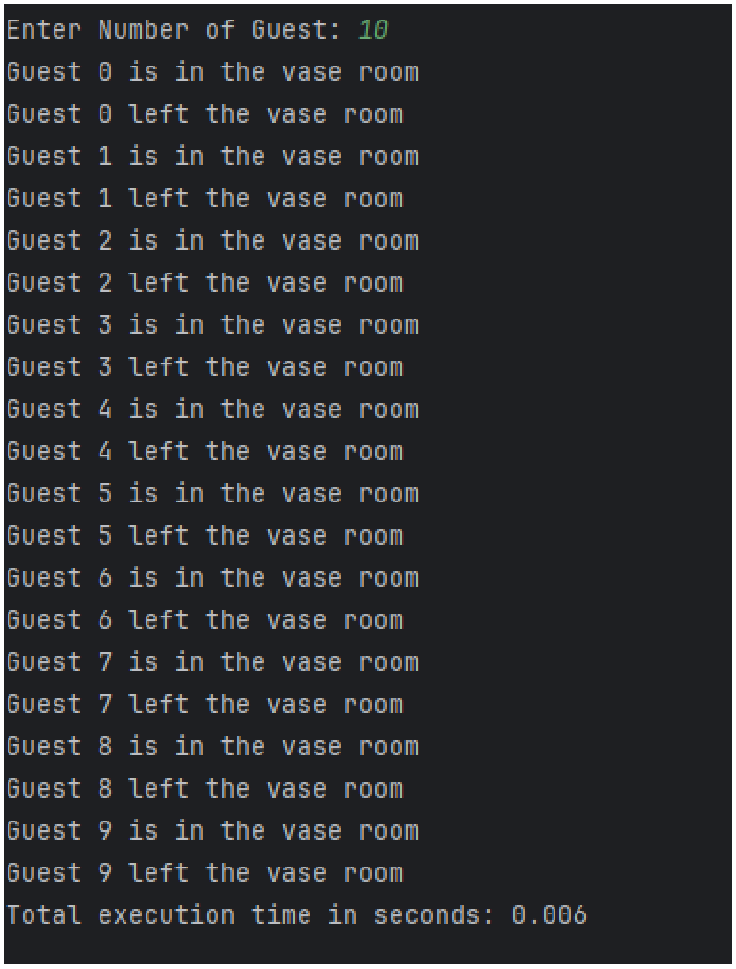
While strategy 1 lets multiple threads in the labyrinth at once the guest could gather around the door and essentially all be collectively checking the door, since guest can go through multiple times and there is no que system some arbitrary guest may repeatedly be let through the door while all other guest wait on them to go in and out of the room over and over again.

Strategy 2:

This strategy is a reentrant lock that allows people to mark the door as busy so while the room is busy, they can come in and do other things while they are trying the lock. This is the optimal solution

Strategy 3:

The queue has the downside that people while in the queue cannot do anything else around the party besides wait for the vase room to be open. While ensuring everyone gets a chance to enter the vase room, they would spend most of their time waiting for the vase room to open.



```
Enter Number of Guest: 10
Guest 0 is in the vase room
Guest 0 left the vase room
Guest 1 is in the vase room
Guest 1 left the vase room
Guest 2 is in the vase room
Guest 2 left the vase room
Guest 3 is in the vase room
Guest 3 left the vase room
Guest 4 is in the vase room
Guest 4 left the vase room
Guest 5 is in the vase room
Guest 5 left the vase room
Guest 6 is in the vase room
Guest 6 left the vase room
Guest 7 is in the vase room
Guest 7 left the vase room
Guest 8 is in the vase room
Guest 8 left the vase room
Guest 9 is in the vase room
Guest 9 left the vase room
Total execution time in seconds: 0.006
```