Documentation

Proof Of Correctness:

Q1:

At first I didn't quite understand the problem in question one, but once I delved deep into the implementation, I understood this was a locking problem. Each guest enters the labyrinth, if they haven't eaten cake before they "lock" the "cake lock" that I implemented. The guests can't communicate with eachother, but to be sure everybody's entered the labyrinth I created a leader guest (guest 7). The only person who can request a new cake is the leader. Note that the guests will only eat the cake one time, no more, thus, when the leader guest has requested a new cake 9 times (there are 10 guests), he knows every guest has entered the labyrinth, and the party can end.

Q2:

Question 2 was a bit more straightforward (I think...), I was able to deduce the most optimal strategy was strategy 3 with the queue (further explanation found under "response to problem 2"). Using this strategy I implemented a queue data structure from java.utils, and loaded in all 10 of my guests. Guests enter one at a time, and once they're done they alert the person in the front of the line (aka the person behind them), that they're good to go. If the guest wanted to re-enter the line (decided by a random), they'd be added to the back of it (queue's are FIFO - First in First out).

Efficiency:

Q1:

It's hard for question 1 to be "efficient" as the order of which guests enter the labyrinth is picked randomly by the minotaur (represented by a random object in mine). Because of this, certain guests might enter the labyrinth many times over before the party can end - making things "inefficient". However, the strategy of having a leader as discussed in the "proof of correctness" has certainly been a more efficient route than just letting guests enter over and over for a set amount of time, then assuming everybody's entered based on luck.

Q2:

Q2 is definitely more efficient than Q1, because of the queue implementation we were able to ensure that guests got in in the order they were in line. However, because guests could re-enter the line (decided by a random) the difference in efficiency is negligible.

Experimental Evaluation:

Q1:

Thankfully I read the first chapter of the textbook, so the leader strategy was immediately apparent to me, thus, this was the only strategy I experimented on. I do believe it'd be inefficient to let the guests enter for a set amount of time, and just assume they've all entered... so I didn't experiment on that strategy.

Q2:

Once again, the strategy was apparent from the beginning for question 2. Thanks to the specific strategy declarations from the question I was able to specifically implement a queue, load the guests into it, and send them into the showroom one at a time to keep the vase safe. No other experiments/strategies were required.

Response to Problem 2

The Minotaur decided to show his favorite crystal vase to his guests in a dedicated showroom with a single door. He did not want many guests to gather around the vase and accidentally break it. For this reason, he would allow only one guest at a time into

the showroom. He asked his guests to choose from one of three possible strategies for viewing the Minotaur’s favorite crystal vase:

1) Any guest could stop by and check whether the showroom’s door is open at any time and try to enter the room. While this would allow the guests to roam around the castle and enjoy the party, this strategy may also cause large crowds of eager guests to gather around the door. A particular guest wanting to see the vase would also have no guarantee that she or he will be able to do so and when.

2) The Minotaur’s second strategy allowed the guests to place a sign on the door indicating when the showroom is available. The sign would read “AVAILABLE” or “BUSY.” Every guest is responsible to set the sign to “BUSY” when entering the showroom and back to “AVAILABLE” upon exit. That way guests would not bother trying to go to the showroom if it is not available.

3) The third strategy would allow the quests to line in a queue. Every guest exiting the room was responsible to notify the guest standing in front of the queue that the showroom is available. Guests were allowed to queue multiple times.

Question:
Which of these three strategies should the guests choose? Please discuss the advantages and disadvantages.

Answer:

Strategy #:	**Advantages:**	**Disadvantages:**
#1	- Easy Entry to room (attempt - if available go in, if not stay out).	- No mechanism to keep order of people who should be entering.
	- Not complex/hard to implement.	- No certainty that a guest who wants to enter will be able to - somebody could HOG it.
		-Wasted Resources/Time when guests try to enter but room is not available.
#2	- Guests don't have to attempt to enter room if it's busy, they can just check the sign - saves resources/time.	- No mechanism to keep track of order - guests may be skipped/cut or miss out.
	- Not Complex/Hard to implement.	- Guests have to remember to mark sign as Available when exiting and BUSY when entering (might be expensive).
#3	- Order of line is kept - all guests who enter the queue will eventually be able to get in.	- Guests have to interact with the guest behind them (save info of one behind - might be expensive).

	- Guests won't get skipped in line.	- Cost & Efficiency of keeping track of queue may be expensive overall.

Based on the observations I've made in the table above discussing the disadvantages and advantages of each strategy, the guests should choose strategy 3. As we can see, the first two strategies don't keep track of an "order" of guests entering the room. Without an order, things can get chaotic... What if one of the guests is super strong and fights every other guest, hogging the room/vase for himself. With the queue discussed in strategy 3 there's much more order, and every guest who enters the waiting queue will at least be able to know that they'll be entering the room eventually to see the vase (depending on their position in the queue).