Programming Assignment 2

Note 1:

Please, submit your work via Webcourses.

Submissions by e-mail will not be accepted.

Due date: Thursday, February 23rd by 11:59 PM

Late submissions are not accepted.

Note 2:

This assignment is individual.

You can use a programming language of your choice for this assignment (limited to C++, Java, and Rust).

If you do not have a preference for a programming language, I would recommend C++.

Problem 1: Minotaur's Birthday Party (50 points)

The Minotaur invited N guests to his birthday party. When the guests arrived, he made the following announcement.

The guests may enter his labyrinth, one at a time and only when he invites them to do so. At the end of the labyrinth, the Minotaur placed a birthday cupcake on a plate. When a guest finds a way out of the labyrinth, he or she may decide to eat the birthday cupcake or leave it. If the cupcake is eaten by the previous guest, the next guest will find the cupcake plate empty and may request another cupcake by asking the Minotaur's servants. When the servants bring a new cupcake the guest may decide to eat it or leave it on the plate.

The Minotaur's only request for each guest is to not talk to the other guests about her or his visit to the labyrinth after the game has started. The guests are allowed to come up with a strategy prior to the beginning of the game. There are many birthday cupcakes, so the Minotaur may pick the same guests multiple times and ask them to enter the labyrinth. Before the party is over, the Minotaur wants to know if all of his guests have had the chance to enter his labyrinth. To do so, the guests must announce that they have all visited the labyrinth at least once.

Now the guests must come up with a strategy to let the Minotaur know that every guest entered the Minotaur's labyrinth. It is known that there is already a birthday cupcake left

at the labyrinth's exit at the start of the game. How would the guests do this and not disappoint his generous and a bit temperamental host?

Create a program to simulate the winning strategy (protocol) where each guest is represented by one running thread. In your program you can choose a concrete number for N or ask the user to specify N at the start.

Problem 2: Minotaur's Crystal Vase (50 points)

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.

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

Implement the strategy/protocol of your choice where each guest is represented by 1 running thread. You can choose a concrete number for the number of guests or ask the user to specify it at the start.

Grading policy:

General program design and correctness: 50%

Efficiency: 30%

Documentation including statements and proof of correctness, efficiency, and experimental evaluation: 20%

Submission:

You will submit a link to your GitHub page. Your repositories **must be private** until the next morning. **You must still push your code before the deadline, because Github will record this time. No code pushes will be accepted after the deadline.** However, we won't start grading until the next morning.

If we cannot access your repositories, or if you provide an invalid link, you will receive a 0 - *please double check your submission once it has been made*.

Late submissions will receive a 0 as per the syllabus.

This GitHub management does two good things:

1. Removes the temptation to look at other's work, because they will all be private until after the deadline.

2. Makes your life easier, as you don't have to send us invites

Also, we cannot accept any late work as all the repos will be potentially public shortly after the deadline.

Happy coding!