CS273: Final Code Review

# Questions to answer:

1. Who did you work with?

Michael Gamlme

1. How is time defined?

Time is defined as one cycle of a for loop represents a minute in time of the simulation. The total time was set to a week which is 10080 cycles of a minute.

1. What kinds of things can happen at each “moment”?

For each moment first the simulation checks if any more patients are entering the waiting room of the hospital. After that the step there is a check if any nurses or doctors are busy and if they are not busy gives them a patient from the waiting room. Finally, the simulator treats the patients assigned to the doctors or nurses. If the patient is fully treated, then they leave the hospital and their record is stored.

1. What happens when the parameters of the simulation changes? Do the results make sense?

When the number of patients per hour increases and the number of doctors and nurses remains the same the wait time increases. When the number of doctors and nurses increase the wait time decreases. Both of these situations follow the logic of what the simulator should complete.

1. Were there any bugs or issues that you fixed?

Using iterators to reference information that was being displayed was not functioning properly. The bug turned out to be that pointer was not referencing properly. There was a bug in calculating the average wait time of a patient. This was caused by dividing the total wait time by an incorrect value when doing calculations.

1. Explain how each data structure is used.

A vector was used to hold all of the names of all possible patients from a text file. When a new patient arrives a name is taken from that vector and a Patient object is created and stored in a priority queue. When the patients have been treated they are removed from the priority queue and moved into a multiset that stores a pointer to a Patient object so they can be found and referred to later.

1. What alternative data structures were considered? Why were they not chosen?

A map, multimap, or hash table were considered to use instead of a multiset to store records of treated patients. They were not used because of time it would create increased complexity and take more time to complete implementation in the project.

1. What alternative data structures would you have also considered?

I might have used a map to store patient record instead of a multiset. I felt that a map was easier to use to find and then reference Patient objects based on their name. Most of all the other data structures were the same ones that I used in my project.

1. Does the class hierarchy make sense? Are there other refactoring opportunities?

There is a class called EmergencyRoom that doesn’t seem to do anything but the solution doesn’t build without it so fixing that problem would help make the class hierarchy make more sense. The Simulator class is very large and has opportunities to be broken up into small pieces to improve readability of the code.

1. Is memory managed correctly (e.g. are pointers correctly deallocated)?

The memory management appears to be solid. It appears that all pointers are cleaned up when not being used and I don’t see any problems.

1. Are exceptions handled correctly?

Invalid or out of range exceptions from user input are handled well by allowing trying again and displaying the problem. No exceptions are handled besides this so if an incorrect argument is passed or returned there is no way of stopping the program from crashing.