Nate Esrey

Code Review

0. Who did you work with?  Sarnell and Isabella

1. How is time defined?

Time is an int, for loop that counts for an amount of minutes and updates the simulation every minute on the clock.

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

Each moment starts through the wait room based on the arrival rate, patient objects created that adds patients, creates random priority based on percentages, checks available nurses first because nurses treat faster, moved down the line until all caretakers are done, when done, persons and caretakers are thrown into a map, go through a treating map based upon the random treatment, kicked out of treatment room when done being treated, then doctors and nurses released, map of name and patient object created for the record, queue of priorities to keep track of past queues, update occurs every minute

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

Less Doctors and nurses, two different levels of exception handling but unable to hold back the user from continuing (the project does not stop when false information is provided), no results due to inability to get user input name when look up the record

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

One main issue was with the record. The patient name was unable to be called due to a possible string error, integer and double issues occurred, pointers and “having them speak to each other” to give them access before updates, iterations through treating map had issues because once the patient was gone, there was no way for the iterator to get the name back (iterations through treating)

5. Explain how each data structure is used.

Queues of doctors and nurses, of all patient names, of each patient object of past priorities

Maps for linking doctor to patient and for records (keys are names), link name of patient object to keep track of records

Priority queue of patients

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

Considered linked list to randomly accessed names, chose priority queue due to simplicity

Considered vectors but did not want to deal with size and changing each element

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

Also considered linked list to randomly fetch a name

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

Yes, caregivers: nurses and doctors, patients and caregivers could be people, name , age, etc that all people have to include patients and doctors, have one higher hierarchy for all people in the hospital

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

Morgue function created to kill off all of the patients once they are finished and leave the E.R, all 2000 patients die in this case, E.R fails

10. Are exceptions handled correctly?

Yes, unit testing was made, try catch made, setting caregivers and setting arrivals, attempted with zero and negative numbers, all worked successfully