Nate Esrey

Kelly Olivier

Hospital Simulation Final Specs

Requirements Specifications:

* Create a working Hospital Simulation for the residents of CS273ville that is efficient as possible, maximizing all functions of a hospital in the quickest way possible.
* Patients that come into the Emergency Room are organized based upon their condition, one to ten in one queue and ten to twenty in another queue.
  + Doctors are able to take one to twenty, but are prioritized on those with conditions over ten. Nurses can only treat those with one to ten conditions.
  + Nurses take up to ten minutes with a patient while doctors take up to twenty minutes with treatment.
* Simulation in the E.R runs one week time or 10,080 minutes.
* User is allowed to input the rate at which patients enter the E.R with the number of doctors and nurses currently working the E.R at that time
* Simulation will: calculate the average patient wait time and be able to hold a record of each patient from cs273ville, allowing the user to go back and search a name with the patients’ past visits.

Pseudo-Code Program:

prompts user for number of patients per hour

User: user inputs number of patients

User: enters inputs out of range

Program: throws exception

Program: prompts user for number of doctors per hour

User: enters inputs out of range

User: inputs number of doctors

Program: throws exception

Program: prompts user for number of nurses per hour

User: inputs number of nurses

User: enters inputs out of range

Program: throws exception

Program: shows stats, and prompts user to choose from menu.

User selects option number one

User selects out of range

Program clears input and displays menu again

Program displays all patients treated, shows menu again

User selects option number two

Program prompts user for patient name

User enters patient name

User enters invalid name

Program: throws exception

Program displays that patients record, displays menu again

User: selects option three

Program ends the program

EmergencyRoom::medicsFree()

Test to identify if a doctor is free

If not free, return false

If free, return true

EmergencyRoom::doctorsFree()

Use number of doctors that the user has entered

Test to see if there is a free doctor that is not being used from doctor queue

For number of medics available, determine which ones are doctors

Increment number of doctors

If a doctor is busy, increment test to show that doctor is busy

If test = number of doctors, return false

Else, there are free doctors available

EmergencyRoom::updateDoctors()

For all unsigned number of medics that are in the E.R

If all medics are busy

Update clock, record the patient that was treated, how long they were in the E.R, and other record information that needs to be recorded

While

Patients over 10 and under 10 are in the E.R queue

Check for free medics

Assign patients that are free into the respectable priority queues, whether that is doctor or nurse.

If patients in queue that do not have over 10 priority

Assign patients to medics since doctors that are free can take under 10 only if there are no over 10 patients

EmergencyRoom::EmergencyRoom(int numDoctors, int numNurses, int arrivalRate) : arrivalRate(arrivalRate), numTreated(0), totalwait(0), clock(0))

Build database for emergency room that includes user input and calculations from the random generator

Create new nurse and new doctor for the numbers input by user

EmergencyRoom::Update()

Update clock

If new patient come in

Place patient in the correct queue based upon priority level and time spent in Emergency room waiting

Increment the visit count for that patient to be “saved” in records

Update doctors()

EmergencyRoom::ListNames()

Iterate through names from CS273ville

Keys for the iterator are the names of individuals

EmergencyRoom::getRecord

Map

If name is not it, no records found

Else, name found

Output patient name, number of visits, and illness priorities.

Use Cases:

*EmergencyRoom:*

|  |  |
| --- | --- |
| *Function name/class name* | *Use for specific code* |
| 1. *MedicsFree ()* | *1. Determines the number of qualified medics in the E.R and if they are busy or free* |
| 1. *DoctorsFree()* | *2. Tests to identify if there are free doctors or not based upon the user’s input of number of docs in E.R.* |
| 1. *UpdateMedics()* | *3. Updates and checks if any medic has completed his or her patient treatment*  *- if completed, update records on that treated patient*  *- assign new patients to medics based upon priority level* |
| 1. *EmergencyRoom()* | *4. Creates the database for the entire program and takes in input with calculations from random generator* |
| 1. *Update()* | *5. Add a new patient if there is a free medic based on priority, update clock* |
| 1. *ListNames()* | *6. List the names in the E.R and iterates through the record* |
| 1. *GetRecord()* | *7. Searches the record by name of patients, outputs patient records if a name is found* |

*RandomGenerator:*

|  |  |
| --- | --- |
| *Function Name/Class name* | *Use for specific code* |
| *1. Build Database ()* | *1. Gets all names from the CS273ville text file and stores such names into the vector of names* |
| *2.RandomPriority()* | *2. Generates a random illness from 1-20 for a patient that “enters the E.R.”*  *70% are 1-10, 20% 11-15, 10% 15-20* |
| *3. RandomSurname()* | *3. Get a random name from cs273ville file* |
| *4. NeedNewPatient()* | *4. get a new name for a patient if a new patient needs to come in based on the rate entered* |

*Main:*

|  |  |
| --- | --- |
| *Function Name/ Class Name* | *Use for Specific code* |
| *Int main()* | *Welcomes the user to the E.R. and prompts the user for input of number of doctors, number of nurses, and the rate at which patients enter the E.R*  *Simulates the program to run*  *Calculates the average wait time for the E.R*  *Includes the Record for the User to search through by name* |

*Patient:*

|  |  |
| --- | --- |
| *Function Name/Class Name* | *Use for Specific Code* |
| *bool Patient::operator<(const Patient Other) const* | *Compares illness priority of each patient that is randomly generated. If priorities are the same, then the comparison switches over to time of when the patients came into the E.R as a “Tie-Breaker”*  *Sorts patients in their respective priority queues* |
| *Class Patient {}* | *Creates the Patient that every individual will have when he or she walks into the E.R.*   * + - *name*     - *arrival time*     - *priority illness number*     - *departure time when he or she leaves the E.R* |

*Physician:*

|  |  |
| --- | --- |
| *Function Name/Class Name* | *Use for Specific Code* |
| *Class Physician {}* | *Creates the Physician for the E.R*  *Each Physician has a current patient that he or she treats*  *Current Treatment time stores the time required to treat the patient based upon the patient’s priority number*  *Number of doctors are input by the user, which affects the rate at which patients are treated* |
| *Class Doctor : public Physician {}* | *Contains a default constructor that will add in a new patient when a new patient is available to be treated by a doctor* |
| *Class Nurse : public Physician{}* | *Inputs a new patient into nurse queue for an available nurse to treat a patient under a priority level of 1-10* |

*Random Generator:*

|  |  |
| --- | --- |
| *Function Name/Class Name* | *Use for Specific code* |
| *Void BuildDatabase(std::vector<std::string> &names)* | *Gets all names from the text files and stores them to the vector of names*  *What runs the entire program and makes it “clean” and efficient* |
| *Int RandomPriority();* | *Generates the random illness to the name from the text file*  *Conducted Randomly* |
| *Std:: string RandomSurname (std::vector<std::string> &names );* | *Picks a random name from the CS273ville list* |
| *needNewPatient(int ratePerHour)* | *Uses the inputted rate to randomly decode if a new patient needs to be generated*  *Done “on the fly” when patients are discharged and brought in depending upon the rate.* |

*UML DIAGRAM:*

(Include UML here)