Requirment Specification:

The hospital simulator is a program that simulates a hosipital primarly the emergency for one week. The simulation can run a varity of scenarios depending on what the user inputs. The limitation for the simulation is a maximum of 60 patients per hour and 60 avalible nurses and doctors. The simulation will begin by uploading all of the citiznes names into two separate ectors one for First name and one for Ser name. after the 2000 first names and 1000 last names are in the program, it will beign to ask the user to input the information about the hospital. The program will ask the user three questions. This part of the program is ran in the EnterData() function in the EmergencyRoom() class

1: Number of Patients Per hour

2: Number of Doctors

3: Number of Nurses

After each question the user will input the a response the program will make sure the response is valid and within the range of options.

After the user enteres the data the program will then send the information the waintingRoomQueue() along with that all three queues (waitingRoom, treatmentRoom, dischargeRoom) will be sent the needed rooms information so that patients doctors and other information can be passed along from one to the other. After all information is in the right place the program will then start the runsacredHeart() function in this function there is a for loop that increments 1 minute at a time for one weeks time. In the for loop there are three other function in here each one responsible for a differnet queue.

To start the loop is will go inside the waiting room queues update function. In here it will randomly select a double and if that number is less than the rate inputed in the begging of the program then a new patient will be added. Depending on the patients illness level (random % 20 + 1) and the capacity of the queues the patient will either be added to he doctors or nurses queue. Along with the patient being added to a queue the program will look to see if the patient already exist in the hospital records by looking up its first name if the patient already exist then it will add to the patients record but if the patient doesn’t exsit the it will create a new patient recored. Ocne the patient is either added or not addd the wating room queue is complete.

Next the treatmentRoomQueue’s update function will be called insed of the treatment room queue it will update the doctorsQueue first. Inside of the doctors queue it is going to check if any patients are being treated by doctors. If true then point to the patient that has been there the longest and see if they have been there long enough to be fully treated. If true then remove that patient from the treatment room and tell the waiting room that there is one more avalible doctor. Next it is going to check to see if there are any availbe doctors left if there is then, then it will start trying to add patients form the wainting room to the treatment room. When it adds patients to the treatment room it will let the waiting room know that thre is one less available doctor and then remove set the patients start treatment time to the current time. After it finishes updating the doctors queue it will do the same exact thing for the nurse’s queue.

After treatment queue the it will go inside of discharges update() function. In this function it will start off by checking to see if there are any patient ready to be discharged. If there are any the program will point to the front of the queue and set its discharge tie and then send the patient on its way. Once the patient is removed the program will update the total visit time for all of the patients so far along with that it will update the number of patients treated. Finally it will delete the patient pointer. Once it is don’t checking if any of the doctors patients are ready to be treaters it will do the same exact thing for the nurses.

After it completes this cycle it ill repeat that 10080 more times unitl it has finished simulating. Once complete. The program will move on to show stats a function inside of sacred heart.

In here the program wil inform you of how many patients were treated and what the average visit time was. After the program informs you of that it will prompt you to select from the menu. In the mune thre is three options

1: list patients

2: find patient records

3 end the program

If the user selects list patients then the program will retrieve all the patient records and out put each name from the records.

If the user selects patient records then the the program will prompt the user for the patients name and once done the program will retrieve all information for the user if that patient exist.

If the user ends the program the program will end.

***UML...***

Use CASES:

Program: prompts user for num of patients per hour

User: user inputs num of patients

User: enters inputs out of range

Program: throws exception

Program: prompts user for num of doctors per hour

User: enters inputs out of range

User: inputs num of doctors

Program: throws exception

Program: prompts user for num of nurses per hour

User: inputs num 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

enterInfo() {

1 add people to vector

2 input patients per hour

3 divided input by 60 (per hour)

4 input num of docs

5 input num of nurses

6 go to in waitingroom queue and pass through patPerHour in setRate(double)

7 go to in waitingroom queue and pass through numOfNur in setNur(Nurses)

8 go to in waitingroom queue and pass through numOfDoc in setDoc(Doctors)

9 go to in waitingroom queue and pass through the vector in Citizens class in setFirstName(vector<string>)

10 go to in treatmentRoom queue and pass through the dischargeRoom Queue pointer in setDischargeeRoomQueue(DischargeRoomQueue \* d)

11 go to in treatmentRoom queue and pass through the waitintgRoom Queue pointer in setWaitingRoomQueue(WaitingRoomQueue \* d)

12go to hospitalrecords and pass through the hospitalrecords pointer in setHospitalRecords(hospitalRecords \* h)

}

runSimulation() {

1 for each minute In a week

2 go to waitingRoomQueue and update wainting room queue

3 go to treatmentRoomQueue and update treatment room queue

4 go to dischargeRoomQueue and update discharge room queue

}

2a

waitingRoomQueue->update(int clock)

{

1 update random location in vector

2 update random patient’s illness level

3 update random treatment time for doctors

4 update random treatment time for nurses

5 if random number is less than patPerHour

{

5a Create temp patient record

5b if(there are docs or nurs avaible)

{

5b.1 If(if illness lvlfits docs needs)

{

5b.1.1 add patient to doc queue

` if(patient name is not a copy)

5b.1.2 add patient info to patient records

}

5b.2 else if(fits nurs need)

Add patient top nurses queue

If(name is not a copy)

5b.2.1 add patient info to patient records

}

}

}

3a

TreatmentQueue(int clock)

{

0 open up docUpDate()

1 if queue is not empty

2 then check for patients that are don’t being treated

3 if any patients then point to the patient in the front of the queue

4 remove patient from queue

5 add doctor to waitingroom

6 set discharge time to clock

7 if waiting room queue has patients

8 while there are doctors and and waiting patients

9 point to patint in waiting room queue

10 remove patient from waitingroom

11 take away one doctor

12 set patient start timem to clock

13 add patint to doc queue

0 open up nurUpdate()

1 if queue is not empty

2 then check for patients that are don’t being treated

3 if any patients then point to the patient in the front of the queue

4 remove patient from queue

5 add nurse to waitingroom

6 set discharge time to clock

7 if waiting room queue has patients

8 while there are nurses and and waiting patients

9 point to patint in waiting room queue

10 remove patient from waitingroom

11 take away one nurse

12 set patient start time to clock

13 add patint to nur queue   
}

4a

DischargeRoomQueue->update(clock){

1 if doc queue is not empty

2point to front patient

3 set discharge time to clock

4 remove patient

5Add visit time to total visit tie

6 increment total number of patients

7 delete patient pointer

8if nurseQueue is not empty())

9 point to front patient

10 set discharge time to clock

11 remove patient

12Add visit time to total visit tie

13 increment total number of patients

14 delete patient pointer

}

5 showStats()

{

1 output num of patients treated

2 out put avg time to treat patients

3 display menu

}

6 statsMenu()

{

1 cin choice

2 open switch statement

Case 1: listPatients()

{

Output all the patients name

}

Case 2: findPatient()

{

Cin patient name

Output patients hospital records

}

Case 3: endProgram()

{

Return 0;

}

}

Project Summary

That includes answers to the questions given below in the project descriptions

~My avg visit time doesn’t work. Regardless of the inputs the avg treatment time is around 1300.

Also indicate what was changed from your initial design in your final implementation.

~At first a lot changed from my initial design in the begging I was using primarily queues and vectors. Then towards the middle end I ended up switching to maps. Somewhere along the way I broke the program and coulndt find where the actual bug was so I ended up starting over this time I started with maps and didn’t get that far I realized that queues were what at least seemed to be the best way to go. I ended up redoing the project again this time with queues. So from the very begging to the very end the overall concept didn’t change much outside of utilizing friend classes a little more and using inheritance a little less. But there was a lot of drastic changing in the middle.

~my implementation did not work. My design felt pretty simple The idea was that the user would enter information about the simulation the I would start in the waiting room queue create a patient and then go into the treatment queue bring a patient from previouse queue and treat patient. Then I would go in thee discharge queue and if there was a patient then I would add its information and the send the paitent on its way. I think the biggest challenge was keeping track of all of the pieces after long periods of time and accounting for all of the potential effects after just changing or adding or removing a certain part of the code. I think that is where a lot of my bugs cam from. Also why I struggled so much to find all of the bugs. Even though I wrote all of the code and it made sense when I wrote it and it made sense in the scope of what it was doing when I got towards the end it was difficult to pinpoint where and why a bug was occurring. Especially ones that don’t actually ruin the code and just output the wrong information. I learned a few lessons throughout this project one of them was one that I have been learning throughout the whole year, and that lesson is to think through function or a problem in the code instead of just going in head first and improvising as I go. That with out a doubt would have saved me so many hours with this project. Another lesson I learned was to not delete anything something I have started to try and do is to create a extra file and move all my big mess up code in there that way I can refer to the good parts of those function in the future. I don’t know if that is good practice or not but it seems to help.