MIDDLE EAST TECHNICAL UNIVERSITY, NORTHERN CYPRUS CAMPUS
CNG213 Data Structures – Programming Assignment 2

Date handed out: **09 December 2020,**
Date submission due: **23 December 2020, 23:55 (Cyprus Time)**

Emergency Room Simulator

The emergency room is one of the most vital parts of any hospital. When a patient arrives, the head nurse first checks the patient and assign him/her to be seen by a doctor. In this hospital, the head nurse will have an app on special handheld devices that they can use to assign patients to doctors in order of emergency. There may be many patients who come to the emergency room, and therefore these doctors need to see the patients and provide treatment in a **fair manner**.

In this assignment, your task is to simulate the doctors serving the patients to gather some statistics. These statistics will be used to check the quality of health care services. Overall, the main goal of this assignment is to help you practice queue and priority queue abstract data types.

In this emergency room, there is a single queue, and all patients who arrived at the hospital are located in a queue to be seen by the doctor(s). Table 1 shows the emergency type with the priority values of the patient where the highest priority is 4 and the lowest priority is 1.

Table 1. Patients Types

Colours code	**Meaning of the colour**	**Priority**
Red (R)	Emergent	4
Orange (O)	Urgent	3
Yellow (Y)	Less Urgent	2
Green(G)	Non urgent	1

You will need to create and maintain a priority queue for the patients. They will be seen by the doctor based on their priority and also their arrival time. Let the following queue be the current queue of this emergency room.

Front | R | Y | Y | G | G | Rear

When a new patient comes to the emergency room, and this patient has a situation that is considered less urgent (Yellow code). Then, s/he is located in the fourth position of the queue (after the first two less urgent patients). This is because his/her priority is less than patients who need immediate resurrection or urgent attention. Therefore, the queue will be as follows:

Front | R | Y | Y | **Y** | G | G | Rear

First of all, you need to create a list of patients that will arrive to the hospital. For each patient, you need to keep the track of the following:

- Arrival time (in minutes, such as at 15th minute) – this will represent when patient arrived at the hospital;
- Service time (in minutes) – this will represent how long the patient will be served/seen by the doctor;
- Service start time (in minutes) – this will represent the actual time the doctor will start to serve/check this particular patient;
- The ID of the doctor who is handling the patient (the ID of the first doctor is 1, the ID of the second doctor is 2, … so on) – This is to be able to represent the possibility that the emergency room might have more than one doctor.
- Insurance – this will represent whether or not the patient has insurance or not. The patient insurance can be one of the following: 0 (insurance is not available) or 1 (insurance is available).

The number of patients, the number of doctors, the maximum arrival time, and the maximum service time should be provided as an input at the beginning of the program.

You need to name your program as **`EmergencyRoomSimulator`** and needs to run on a command line by using the following format:

`EmergencyRoomSimulator` `noOfPatients noOfDoctors maxArrivalTime maxServiceTime`

An example of a patient list created with the following input is shown in Table 2.

`EmergencyRoomSimulatorSimulation` `5 2 20 5`

Based on the given input, there are 5 patients in the hospital who need to be seen by the doctor. The maximum arrival time is 20 and the maximum service time is 5. Once you get this input data from the user, then you need to prepare your simulation data based on this given input. At the beginning, the fields of service start time and doctor ID are equal to 0, because you need to update these fields after the patient is seen by a doctor. Other fields will be prepared as follows: Since in the given example, you have 5 patients, you need to prepare simulation data for 5 patients based on the parameters given above. Please see example data prepared in Table 2. You need to use a randomiser for arrival time and service time generation. However, you need to remember that the arrival time needs to be less than maximum arrival time given by the user and the service time needs to be less than the maximum service time.

You also need to randomly generate a type for patients, which can be red (R), Orange(O), Yellow (Y), and Green (G). You also need to randomly assign a patient insurance. You can see an example data in Table 2 which is prepared based on the input above.

Table 2. An example of a patient list

Patient 's Type	Arrival Time	Service Time	Service Start Time	Doctor ID	Insurance
R	5	0	0	0	0
Y	7	5	0	0	1
Y	8	5	0	0	1
G	15	10	0	0	0
G	20	12	0	0	1

When your patients' list is ready, you need to create an empty queue and an integer array to keep the availability of the doctors (0: Busy, 1: Available). For example, if there are two doctors, you need to have an array of 2 integers. As these doctors are available at the beginning, you need to initialise the array with 1s

Once these are ready then you need to start processing the patients which are prepared in the patients' list. When a patient comes to the Hospital at his/her arrival time (note that you generated them randomly), you need to check the current state of the queue.

- If nobody is waiting in the queue, you need to randomly assign the patient to one of the doctors to be seen and you need to make the availability of that doctor "Busy" and then update the service start time for the patient.
- If the queue is not empty, you need to locate the patient in the appropriate position in the queue (see an example above).

When a patient is treated by a particular doctor, the doctor ID should be updated for the patient. After that, the availability of the doctor should be converted to "Available". When a doctor is available for the next patient, and there is another patient in the queue, the patient should be assigned to that doctor. When more than one doctor is available at the same time, you need to randomly assign the first patient in the queue to one of the available doctors. When all patients are treated, the simulation will be completed.

When the simulation is completed, you need to report the following information/statistics:

- The number of doctors: How many doctors served the patients? (This is already given to you by the user)
- The number of patients: How many patients were served? (This is already given to you by the user)
- The number of patients for each patient type: How many patients from red, yellow, orange and green are served by the doctors?
- The number of patients for each doctor: How many patients are served by each doctor?
- The completion time: How long did it take to complete the simulation?
- Average time spent in the queue: What was the average delay in the queue? You need to calculate the total waiting time of all patients in the queue and then divide this by the number of patients you have. This will give you the average waiting time in the queue.
- Maximum waiting time: What was the longest wait time in the queue? You need to find the patient who waited longest in the queue.
- Insurance usage: This will display the total number of patients with or without insurance.

When a patient is served by a doctor, you need to put it back to the list of patients that is created at the beginning of the program. Then you need to use the list to provide the required information/statistics.

Programming Requirements:

In this assignment, you are expected to write the following functions:

- **`parseInput:`** This function should parse the input and set the values of the number of patients, the number of doctors, the maximum arrival time and the maximum service time.
- **`createPatientList:`** This function should randomly create patients based on the input (the number of patients, the number of doctors, the maximum arrival time, the maximum service time). The patients should be stored in a linked list in ascending order based on their arrival time.
- **`initialiseSimulator:`** This function should create an empty queue, and also an integer array to keep the availability of the doctors.
- **`newPatient:`** This function takes a patient from the patient list based on the arrival time and add him/her to the queue.
- **`servePatient:`** This function takes a patient from the queue to be served by a doctor.
- **`reportStatistics:`** This function reports the statistics, such as the average time spent in the queue, maximum waiting time, etc. (see above).
- **`main:`** The main function is responsible to coordinate all the functions, simulated clock and other required operations to run the simulator successfully.

Submission Requirements:

In this assignment, you need to have a header file (queue.h) which includes the major functionality of the Queue ADT. If you will use other ADTs, you need to create a separate header file for each of them. You also need to have a C source file (EmergencyRoomSimulation.c) that includes the main function and other functions. You need put all these files into the "cng213a2" folder and then submit the compressed version of the folder to ODTU-CLASS.

Grading Policy:

Grading Item	Mark (out of 100)
Data Structures	5
Input	5
Function: **`parseInput`**	5
Function: **`createPatientList`**	10
Function: **`initialiseSimulator`**	5
Function: **`newPatient`**	20
Function: **`servePatient`**	10
Function: **`reportStatistics`**	10
Function: **`main`**	25
Submission requirements followed	5

Important Notes:

- Remember to have good programming style (Appropriate comments, variable names, formulation of selection statements and loops, reusability, extensibility etc.). Each of the items above will include10% for good programming style.

- Read rules regarding to assignments from the Syllabus carefully.

- If your code does not compile due to syntax errors, you will automatically get zero.

- If your code includes a variable declaration inside a for loop such as for(int i=0; i<5;i++), you will automatically get zero.

- If your code includes gloable varibles, you will automatically get zero.

Sample run could be as follows:

```
***************Report****************
The number of doctors: 2
The number of patients: 5
Number of patients for each patient type:
      red: 1
      orange: 1
      yellow: 2
      Green: 1
Number of patients for each Doctor:
      Doctor 1:4
      Doctor 2:1
Completion time: 23
Average time spent in the queue: 1.8
Maximum waiting time: 5
Number of patients with insurance: 3
Number of patients without insurance: 2
```