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

Date handed out: **29 November 2020, Sunday**
Date submission due: **13 December 2020, Sunday, 23:55 (Cyprus Time)**

Ventilator Simulation

Let's be futuristic and helpful to our community! During the coronavirus disease 2019 (COVID-19) crisis, Cyprus governmental hospital have run short of ventilators (a life-saving device). Any patient who needs a ventilator might not be able to get one, imperilling their survival. Hence, you will write a smart ventilator simulation program to help the governmental hospital to estimate how many ventilators they will need to respond to this crisis, accounting for increased, uncertain patient arrival rates and lengths of stay.

This assignment aims to help you to practice queue and priority queue abstract data types. You will write a simulator that will help governmental hospital to simulate a queue to gather some statistics.

In a hospital, there is a single queue and all patients are located in this queue when they need a ventilator. There are three types of patients and they have a different level of priority based on their diseases level. Table 1 shows the patient types with their priority values where the highest priority is 3 and the lowest priority is 1.

Table 1. Patient Types

Patient Type	Priority
Severe (S)	3
Moderate(D)	2
Mild (M)	1

You will need to create and maintain a priority queue for the patients. They will be served based on their priority and also their arrival time. Let the following queue is the current queue of the hospital.

Front | S | S | D | D | M | Rear

When a new patient comes to the hospital and has a severe diseases level, s/he is located in the third position of the queue (after the first two severe patients) because his/her priority is higher than moderate and mild patients. Therefore, the queue will be as follows:

Front | S | S | **S** | D | D | M | Rear

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

- Patient type (S, D or M) – this will represent their priority;
- Arrival time (in minutes, such as at 15th minute) – this will represent when patient need a ventilator;
- Service time (in minutes) – this will represent how long the patient will be served by the ventilator;
- Service start time (in minutes) – this will represent the actual time the ventilator will start to serve this particular patient;
- Ventilator ID – this will represent the ID of the ventilator that served the patient (the ID of the first ventilator is 1, the ID of the second ventilator is 2, … so) – This is to be able to represent the possibility that the hospital might have more than one ventilator.
- Patient gender – this will represent the gender of the patient. The patient gender can be one of the following: Male or Female.
- Patient age group– this will represent the age group of the patient. The patient age group can be one of the following: Elderly, Adult or Young.

The number of patients, the number of ventilators, 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 **`ventilatorSimulation`** and needs to run on a command line by using the following format:

```
ventilatorSimulation noOfPatients noOfVentilators maxArrivalTime maxServiceTime
```

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

```
ventilatorSimulation 5 2 20 5
```

Based on the given input, there are 5 patients in the hospital who need a ventilator and 2 ventilators. 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 ventilator ID are equal to 0, because you need to update these fields after the patient is served by a ventilator. 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 Severe (S), Moderate (D) and Mild (M). Based on the patient gender and age group types given above, you also need to randomly assign a patient gender and patient age group. 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 Type	Arrival Time	Service Time	Service Start Time	Ventilator ID	Patient Gender	Patient Age group
S	5	3	0	0	Male	Elderly
M	7	5	0	0	Female	Young
M	8	5	0	0	Male	Adult
D	15	2	0	0	Female	Elderly
S	20	3	0	0	Male	Elderly

* You do not need to directly store the patient gender and age group as text value, you can also do abstraction in your implementation.

When your patient list is ready, you need to create an empty queue and also an integer array to keep the availability of the ventilators (0: Busy, 1: Available). For example, if there are two ventilators, you need to have an array of 2 integers. As these ventilators 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 needs to be served 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 ventilators to be served and you need to make the availability of that ventilator "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 served by a particular ventilator, the ventilator ID should be updated for the patient. After that, the availability of the ventilator should be converted to "Available". When a ventilator is available for next patient and there is another patient in the queue, the patient should be assigned to that ventilator. When more than one ventilator is available at the same time, you need to randomly assign the first patient in the queue to one of the available ventilators. When all the patients are served, the simulation will be completed.

You will also need to maintain a simulated clock which keeps the time of the latest event handled. There are three types of events which are as follows: (1) patient arrival, (2) start service and (3) complete service. At the beginning, the simulated clock should be equal to 0. Regarding the simulated clock, two examples of scenarios are given below.

- If the first event is "patient arrival" at the 5th minute, then you need to advance the simulated clock to 5.
- If the current clock is 21 and the nearest event is "complete service" at the 25th minute, then you need to advance the simulated clock to 25.

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

- The number of ventilators: How many ventilators 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 Severe, Moderate and Mild are served by the ventilators?
- The number of patients for each ventilator: How many patients are served by each ventilator?
- 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.
- Most gender usage: This will display the type of the gender that served mostly by ventilators.
- Most group usage: This will display the type of the age group that served mostly by ventilators.

When a patient is served by a ventilator, 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 ventilators, 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 ventilators, 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 ventilators.
- **`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 ventilator.
- **`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 (ventilatorSimulation.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 entilators: 2
The number of patients: 5
Number of patients for each patient type:
      Severe: 2
      Moderate: 1
      Mild: 2
Number of patients for each ventilator:
      Ventilator 1:5
      Ventilator 2:0
```

```
Completion time: 23
Average time spent in the queue: 1.8
Maximum waiting time: 5
Most gender usage: Male
Most age usage: Elderly
```