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

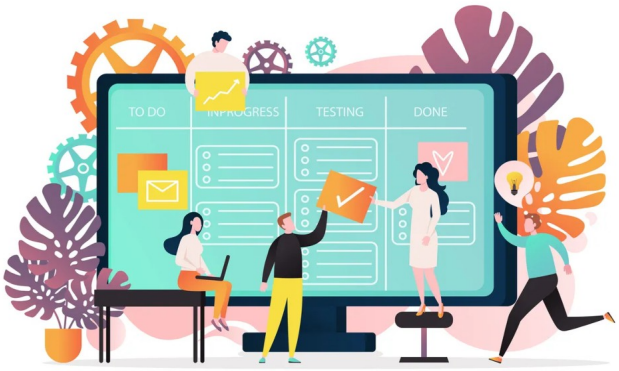
Date handed out: **24 November 2023,**
Date submission due: **8 December 2023, 23:00 (Cyprus Time)**

TODO: Tasks Management Simulator

Purpose:
The main goal of this assignment is to help you practice queue and priority queue abstract data types.

Context:

In computer engineering field, all tech companies frameworks emphasize close collaboration between developers, which then calls for additional non-technical skills like communication, teamwork, and flexibility. In a development project, the number of tasks to be accomplished by the developers is enormous and therefore, task distribution should take a place in a **fair manner**.



In this assignment, your task is to simulate the task distribution and completion process to gather some statistics which will be used to check the team performance and improve software development process. This Tech Company accepts various projects simultaneously. These projects go through different phases and each has a set of tasks that are carefully determined via weekly meetings. To organize the work load among the different projects and through the different phases, the company has a single queue, and **for every phase per project**, **all tasks are located in a queue** to be seen by the developing team. However, tasks are not the same, and each is assigned a label to show how important and urgent it is. **Table 1 shows the labels type with the priority values of the TASK where the highest priority is 4 and the lowest priority is 1**.

Table 1. Tasks Types

Label	Meaning of the label	Priority
C	Critical	4
H	High priority	3
M	Medium	2
N	Normal	1

You will need to create and maintain a priority queue for the **Tasks (no matter to what project it belongs)**. They will be given to the developer **based on their priority and also their arrival time (arrival time may vary because a developer may work on more than one project at a time, and the weekly meeting may occur at different times). To understand this,** consider that the following queue to be the current queue of this Company.

Front | C | M | M | N | N | Rear

After a meeting, the team manager decided to add a new t**ask**, and this **task** has a label that is considered medium (M code). Then, it is located in the fourth position of the queue (after the first two medium **Tasks**). This is because its priority is less than the other tasks which need immediate or urgent attention. Therefore, the queue will be as follows:

Front | C | M | M | **M** | N | N | Rear

To simulate this, you need to create a list of **Task** that will be added to the queue (). For each **Task**, you need to keep the track of the following:

- **Arrival time** (in minutes, such as at 15th minute) – this will represent when the **Task was added to** queue;
- **Service time** (in minutes) – this will represent much time the developer will need to finish a **task**;
- **Service start time** (in minutes) – this will represent the actual time the developer get to the t**ask**; Means at that time, the developer started working on that specific task.
- **The ID of the developer who is handling the task** (the ID of the first developer is 1, the ID of the second developer is 2, … so on) – This is to be able to represent the possibility that the developing team size may vary.

Programming Details:

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

Format: TODO_Simulator noOfTasks noOfDevelopper maxArrivalTime maxServiceTime

The number of tasks, the number of developers in a team, the maximum arrival time, and the maximum service time should be provided as an input at the beginning of the program. An example for possible input is:

Example: TODO_Simulator 5 2 20 200

According to the given input example, there are 5 tasks at the company which need to be accomplished by the development team that has 2 people. The maximum arrival time is 20 and the maximum service time is 200. At the beginning, you have to prepare the simulation for 5 tasks, the fields of service start time and developer ID are equal to 0. Later, you need to update these fields once the task is handed to the developer. For the label of the Task, you need to use a randomise to randomly generate it, and it can be Critical (C), Medium (M), High priority (H), Normal (N). In addition, you need to use a **randomiser** for arrival time and service time generation. Please remember, the arrival time needs to be less than maximum arrival time given by the user. Similarly, the service time needs to be less than the maximum service time. An example of a packages list created with the following input is shown in Table 2.

Table 2. An example of a Task list (TODO list)

Packet Label	Arrival Time	Service Time	Service Start Time	Developer ID
C	5	60	0	0
H	7	50	0	0
H	8	50	0	0
N	15	100	0	0
N	20	120	0	0

When your Tasks' list is ready (refer to table 2 for an illustrative example) you need to create an empty queue and an integer array to keep the availability of the developpers (0: Busy, 1: Available). For example, if there are two developers, you need to have an array of 2 integers. At the beginning, these developers are available therefore, you need to initialise the array with 1s.

Once these are ready then you need to start processing the **Tasks** which are prepared in the **Task**s' list. When a **Task** arrives to the Queue at its arrival time (note that you generated them randomly), you need to check the current state of the queue.

- If no **Task** is there in the queue, you need to randomly assign the task to one of the developers to be taken care of and you need to make the availability of that developer as "Busy" and then update the service start time for the task.
- If the queue is not empty, you need to locate the **Task** in the appropriate position in the queue (see an example above).

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

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

- **The number of developers:** How many developer were in charge of projects tasks? (This is already given to you by the user)
- **The number of Tasks**: How many Tasks were accomplished? (This is already given to you by the user)
- **The number of tasks for each label:** How many task was critical, high priority, medium or normal are delivered by the staff?
- **The number of tasks for each Employee/ Developer**: How many tasks are accomplished by each employee?
- **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 tasks in the queue and then divide this by the number of tasks 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 task which waited longest in the queue or being delayed the most.

When a task is treated by the employee, you need to put it back to the list of tasks that is created at the beginning of the program. Then you need to use the list to provide the required information/statistics. **Please note that the developer/employee can work on one task at a time, no multiple tasks can be carried at the same time.**

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 packages, the number of doctors, the maximum arrival time and the maximum service time.
- **createTasksList:** This function should randomly create Tasks based on the input (the number of tasks, the number of developers, the maximum arrival time, the maximum service time). The task 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 developers.
- **newTask:** This function takes a task from the list based on the arrival time and add it to the queue.
- **accomplishTask:** This function takes a task from the queue to be treated by a developer.
- **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 (**TODO_Simulation.**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: **createTasksList**	10
Function: **initialiseSimulator**	5
Function: **newTask**	20
Function: **AccomplishTask**	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 globale varibles, you will automatically get zero.

Sample run could be as follows:

**********Report**********
*The number of Developers is: 2
*The number of Tasks: 5
*Number of Tasks for each Label:
	Critical: 1
	High priority: 2
	Medium: 0
	Normal: 2
*Number of Tasks for each Developer:
	Developer 1 Accomplished: 4
	Developer 2 Accomplished: 1
*Completion time: 23
*Average time spent in the queue: 1.8
*Maximum waiting time: 5