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

Date handed out: Monday 21 November 2016
Date submission due: Monday 5 December 2016

"Car Wash System Simulator"

In this assignment, you will write a program that simulates the operation of a car wash system that is usually found in a petrol station. The whole purpose of this simulator is to gather some statistics about the efficiency of this garage in washing cars. Especially, it aims to find out how long it takes to serve certain number of cars and the average waiting time for a car to be served. In this system, you will get some parameters from the user to configure this simulator but you will also have to assume that there are at least two car washing machines and there can be unlimited number of cars waiting to be washed.

The program will mainly require three inputs to run the simulation:

- The length of time in hours for this simulation which means the total duration of time for the service that will be provided, such as, 2 hours (i.e., the simulation will simulate the system for 2 hours).
- The maximum time it takes for the washing machine to wash one car, such as 5 minutes (i.e, the maximum service time).
- The number of washing machines, such as 3, please note that there can be minimum of 2 (i.e., the number of service queues).

The following elements are required to run the simulation: a timing loop, a "car to wash" simulator, "process car wash" simulator and "start to wash next car" function.

1. **Timing loop**: This is simply the simulation loop. Every iteration of the loop will be considered 1-minute real time. The loop will continue until the service has been in operation (based on the given input above). When the operating period is complete, however, any waiting cars must be served before ending the simulation. The timing loop has the following sub functions:
    - Determine whether a car arrived (car to wash simulator)
    - Process car wash
    - Start to wash next car

2. **Car to wash simulator**: This simulator will use a random number generator to determine whether a car has arrived to the garage or not. Scale the random number to an appropriate range, such as 1 to 10. The random number should be compared with a defined constant. If the value is less than the constant, a car arrived; if it is not, then no car arrived. For the simulation, set the request level to %50; that is, on average, a car will arrive every 2 minutes (it is a very busy garage). If a car arrives, check if a washing machine is available, if it is then start washing the car with that machine, if it is not, then place it into the shortest queue (for each washing machine you have a waiting queue, and therefore you need to place the car into the shortest available queue). If the queues have the same length, then the car joins the one of the queues randomly.

3. **Process car wash**: If a car wash is active, test whether it has been completed. In order to check if it is completed, you will again need to randomly decide whether a car wash is

completed or not. In order to do that, you need to check the time of the service and make sure that it is always less than the given maximum service time. Therefore, if this random decision says that the car wash is not completed but the duration is more than the maximum service time, then you need to consider that the car wash is completed. If completed, print the statistics for the current car wash and gather the necessary statistics for the end-of-job report.

4. **Start to wash next car**: If the washing machines are idle (means there is no active car wash now), then start a new car wash if there is one waiting in the queue for that washing machine. Please note that starting a car wash must calculate the time that the car has been waiting in the queue. This will show the time that the car has been waiting in the queue.

During the processing, print the data shown in the table below after each car wash is completed (**Note**: you will not get the same results). It shows the car wash number, arrival time (the time the car came to the garage), wait time (how long the car waited in the queue to be washed), start time (when the car started to be washed), service time (how long it took to wash this car), the related washing machine number (for each washing machine, you should assign a number), and queue size (shows the total number of cars in the queue of the relevant machine).

Clock time	Car wash number	Arrival time	Wait time	Start time	Service time	Washing Machine	Queue Size
4	1	2	0	2	3	2	2
6	2	3	2	5	2	2	4

At the end of the simulation, print out the following statistics gathered during the processing: Be sure to use appropriate format for each statistic, such as float for averages:

1. **Total cars washed**: The total number of cars to be washed during the operating time.
2. **Total washing machines**: The number of washing machines that should be specified at the beginning of the simulation.
3. **Total idle time**: Total time during which no cars were washed by all the washing machines (how much time we spent without washing cars).
4. **Total wait time**: Sum of wait times for all cars in all washing machine queues.
5. **Total service time**: Total time spent for washing cars in all queues.
6. **Maximum queue size**: Maximum number of cars waiting in a queue at a point in time during the simulation time.
7. **Average wait time**: Total wait time/total cars washed.
8. **Average service time**: Total service time/total cars washed.

Run the simulator twice. Both runs should simulate 2 hours. In the first simulation, use a maximum of service time of 2 minutes. In the second run, use a maximum of service time of 5 minutes.

Programming Requirements:
In order to implement this simulator, you need to enter the three values required to start the simulator from the command line. These values are: (1) the length time in hours for this simulation, (2) the maximum time it takes for a washing machine to wash a car (the maximum service time) and (3) the number of washing machines (the number of service queues). You need to write the following functions:

- **run_simulator**: This function will include the main timing loop.

- **car_to_wash_simulator**: The request simulator will use a random number generator to determine whether a car has arrived to be washed.
- **process_car_wash**: If a car is now being washed, this functions tests whether this car wash has been completed. If completed, print the statistics for the current car wash and gather the necessary statistics for the end-of-job report.
- **start_washing_next_car:** If there are no active car wash, start a new car wash if there is one waiting in the queue.
- **finalise_report_simulator**: This function will mainly finish the simulator with reporting the required data (see the list above).

Queue Header: You need to create a separate header file that includes all the functionalities of a queue ADT. It is important to separate the functions of your queue ADT from the main application which is the car wash system simulator.

Submission Requirements:
Create a project under the **CNG213_Assignment_2** folder. Separate the major functionality of your Abstract Data Types into header files and put them under the **CNG213_Assignment_2/Project_Lib** folder. Project submission should be compressed version of **CNG213_Assignment_2** folder. If you do not follow this structure, you will loose %10 from the overall grade.

Programming Style Tips!
Please follow the modular programming approach. In C we use functions referred to modules to perform specific tasks that are determined/guided by the solution. Remember the following tips!

- Modules can be written and tested separately!
- Modules can be reused!
- Large projects can be developed in parallel by using modules!
- Modules can reduce the length of the program!
- Modules can also make your code more readable!

Grading:
Your program will be graded as follows:

Grading Point	**Mark (100)**
main function	5 pts
car_to_wash_simulator	35 pts
request_simulator	10 pts
process_car_wash	10 pts
start_washing_next_car	10 pts
Header files (in particular, a queue header is required!)	15 pts
finalise_report_simulator(): Generating report by requesting report function (calculating and displaying statistics properly)	15 pts

NOTE: 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.