Programming assignment #1

Implement a queue data type. Then use it to simulate the evolution of waiting queues, by invoking several instances of the queue data type.

To fix your ideas, assume that we are simulating a check-in operation for an airline company at an airport terminal.

- We have two classes of service: first class and coach. Each class of service has a dedicated queue.
- There are two service stations for first class and three for coach.
- Each service station takes passengers from the corresponding queue; but if a first class service station is free and the queue for coach is not empty then the service station serves passengers from the coach queue.
- Passenger arrival times are random, but subject to average arrival times; for example, we get a first class passenger every five minutes and a coach passenger every two minutes on *average*. Actual arrival times are random.
- Passenger service times are also random, but subject to average service times; for example, first class passengers usually require six minutes of service and coach passengers get on average 2 minutes of service. These are average times; actual times vary. All times are measured in minutes.
- The simulation starts with empty waiting queues and free service stations. At some point in time (usually 40 minutes prior to departure time) the company closes the queues (no more passengers are admitted); then the simulation ends when all the queues are empty and all the service stations are free.

Inputs to the simulation:

- The duration of the check in (make it arbitrarily long, do not worry about it being or not being realistic).
- The coach average arrival rate, and average service rate.
- The first class average arrival rate and average service rate.

Outputs of the Simulation:

- The duration of the simulation (which may be longer than the input parameter, as when checkin closes, there may be passengers in the waiting queues and service stations).
- The maximum length of the queue for each queue.
- The average and maximum waiting time for each queue.
- The rate of occupancy of each service station (percentage of time each station was busy).
- If you want: show the real-time evolution of the queues during the run-time simulation.

Programming Assignment #2

Consider the following expression BNF:

```
<expression>  ::=  <term> + <expression>  |  <term> - <expression>  |  <term>

<term>  ::=  <factor> * <term>  |  <factor> / <term>  |  <factor>

<factor>  ::=  <digit>  |  (  <expression>  )

<digit>  ::=   0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

Using recursive descent, and ***only recursive descent***, scan expressions that adhere to this BNF to build their expression tree; write an integer valued function that scans the tree to evaluate the expression represented by the tree.

There are plenty of clever programs online that you can download to evaluate arithmetic expression tree; if you want zero in this assignment, download one and submit it as programming assignment #2; if you want a grade greater than zero, please follow our instructions. Thanks.

Input:

- A numeric expression adhering to this BNF.

Output:

- Some representation of the expression tree.
- The result of evaluating the expression.

Programming Assignment #3

Use Dijkstra’s algorithm to implement a program that computes the shortest distance between two nodes of an undirected weighted graph. Once Dijsktra’s algorithm has executed, use its output to automatically identify a shortest path in the graph.

Input:

- The graph, given by providing in sequence: the number of nodes, then a sequence of arcs with associated length.
- The source.
- The destination.

Output:

- The shortest distance.
- A shortest path.