**CSCI 301 Moustafa Elsayed**

**Computer Science 2**

**Project 8: Queueing simulation**

**Introduction**

Linked Queues are useful when we want to print the contents of a queue without directly accessing the data inside the queue. In this program, linked queues are created based on the user input. Each queue represents a line of customers and one teller. Then the program generates a transaction time amount and starts to count down from it, when the transaction time becomes zero a new customer is dequeued or served and another transaction time amount is generated, and the loop is repeated.

**Data Structures**

This program uses 18 data structures:

* An Item type “data” in the header file: That is used to hold the input data of the linked queue.
* A Node pointer type “next” in the header file: that is used to point towards the next node.
* A Node pointer type “front” in the header file: that is used to point the first node in the queue.
* A Node pointer type “rear” in the header file: that is used to point to the last node in the queue.
* An in type “count” in the header file: that is used to count the number of items in the linked queue.
* An int type “count” in the main file: that is used to count the number of customers served by getting incremented every time a customer is dequeued
* An int type “entry\_time” in the main file: that is used to hold the entry time of the customer when they get enqueued.
* An int pointer type array “trans\_time” in the main file: that is used to hold the transaction time of every dequeued customer.
* An int “wait\_num” in the main file: that is used to hold how long each customer waited to get dequeued.
* An int type “seed” in the main file: that is used to hold the seed value that will be used for the random number generator.
* An int type “z” in the main file: that is used to hold the number of queues the user wants.
* An int type “prob” in the main file: that is used to hold the probability value of a customer arriving based on what the user enters.
* An int type “maxTrans” in the main file: that is used to hold the maximum length of a transaction type based on what the user enters.
* An int type “dur” in the main file: that is used to hold the duration of the program running based on what the user enters.
* An in type “maxWait” in the main file: that is used to hold the maximum waiting time and is incremented every loop.
* An int type “max” in the main file: that is used to hold the maximum waiting time by comparing its value with the “maxWait” variable. If “max” is less than “maxWait” then “max” holds the value of “maxWait”.
* An int “custWait” in the main file: that is used to hold the total number of customers waiting in all queues.
* A Queue type array “line” in the main file: that is used to hold the line of customers.

**Functions**

There are 6 functions in this program:

* A void type “enqueue()” in the header and source file: that is used to add a new item to the linked queue.
* An Item type “dequeue()” in the header and source file: that is used to return an item from the linked queue.
* An int type “length()” in the header file: that is used to return the number of items in the linked queue.
* A bool type “empty()” in the header file: that is used to return whether or not the linked queue is empty.
* A friend function in the header and source file: that is used to print all the contents of the linked queue.

**The main program**

The program first asks the user for the seed value, then the number of queues, then the probability of a customer arriving , then the maximum transaction time a customer can have, and finally the duration of the program. Then the program dynamically assigns the number of queues to the queue “line” and the int “trans\_time”. Then the program assigns every value in the “trans\_time” array to 0. If all these inputs are valid, a loop starts with the duration of the program as its limit, and inside that loop is another loop with the number of queues as its limit. The program then randomly generates a number and mods it by 100, if that number is less than the probability the user entered then the program enqueues the time of the big loop in the queue. Then if “trans\_time” is equal to 0 and the queue is not empty, the program compares “max” with “maxWait” if “max” is less than “maxWait” then “max” hold the value of “maxWait” , then the program resets “maxWait” to zero, increments the variable “count”, variable “entry\_time” hold the value dequeue from the linked queue, and the variable “wait\_num” is added to and equal to the “time” variable minus the “entry\_time” variable, then the “trans\_time” variable is given a randomly generated value, and if that value is 0 then the program adds 1 to it. However, if “trans\_time” is not equal to 0 then the program increments the “maxWait” variable. If “trans\_time” is greater than 0 the program decrements that variable. Then the program prints the “time” value, the “trans\_time” value and all the items inside the linked queue and the small loop is repeated. After the small loop and the big loop ends the program decrements the value of “count” , and the “custWait” variable holds the returned value of “countWait()” function. Then finally the program prints the longest time a customer waited to be served, and the average ticks the served customers waited, and how many customers are still left in line.