Kathmandu University
Department of Computer Science and Engineering
Dhulikhel, Kavre

A Report on
COMP 202: Data Structures and Algorithms
Mini Project

Submitted by:
Sabin Thapa (54)
Rojan Uprety (56)

Submitted to:
Dr. Rajani Chulyadyo
Department of Computer Science and Engineering

Submission Date:
January 17, 2019

Task: To write a program to simulate a queue of customers in a bank. Customers must be served one at a time and the customer that has been waiting the longest must be served first.

Implementation:

Some of the basic algorithms used in our project are given below:

Algorithms:

enqueue()

Input: QUEUE(a, b), data to be inserted.

Output: Updated queue.

Steps:

1.Create a new node, newNode

2.newNode->info=data

3.newNode->predecessor=first

4.If the queue is empty

newNode->next=first;

first=newNode

5.else

while(predecessor->next)

predecessor=predecessor->next

predecessor->next=newNode

6.size++

7.last=newNode

dequeue()

Input: QUEUE(a, b), data to be deleted.

Output: Updated queue.

Steps:

1. If front == NULL

2. print "Underflow"

3. else

4. Initialize temp = front

5. front = front->next
6. If front == NULL
7. rear = NULL
8. else
9. front->prev = NULL
10 Deallocate space for temp

C++ Source Codes

The C++ source codes are submitted with the labels **miniProject.h(**header file), **miniProject.cpp** (driver file) and **main.cpp** (implementation file). The miniProject.h and miniProject.cpp handle the operations such as creating classes, constructor, destructor, inserting queue etc. The main.cpp includes the queue implementation to simulate the customers in a bank on the basis of their waiting and arrival time.

To stimulate a queue of customers in a bank, either a user has to provide data of the arrival and waiting time of the customers and execute the program to see the results or the data can be provided by generating pseudo random number. Here, we've used the **srand()** function, to generate the arrival time and the waiting time of customers , which initializes the pseudo-random number generator by passing the argument seed. Often the function time is used as input for the seed.

If the seed is set to 1 then the generator is reinitialized to its initial value. Then it will produce the results as before any call to rand and srand.

Parameters:

An integer value to initialize the random number generator. Often the function time is used as input for the seed.

Return value:

There is nothing returned by srand.

Working:

The user can provide the following inputs:

1. Simulation time (in minutes):

   It determines how long the program runs. If you want to analyze the simulation of the first one hour of the bank, then you have to provide the input as 60 minutes and the program generates the arrival time, waiting time, transaction time and the no. of customers via srand function and executes the information in a queue.

2. Number of servers:

   The program isn't just restricted to one server. There can be as much servers as the maximum limit of the server provided in the program, so there isn't necessarily just one line of customers. Customer can choose whichever server that's free so that the queue of customers isn't very long.

The program is designed in such a way that the customer that has been waiting the longest is served first. For instance, if there are two servers in a bank and three customers arrive at the same time, then two of the customers are served by each of the servers and the remaining customer stands in either of the queue. If the transaction time of one customer is greater than the other, the program puts the third customer in the queue in which the transaction time of the customer is smaller.

Output of the above instance from the program:

```
Please input the following data(Time in minutes):

Length of the Simulation(in minutes): 5
Average Number of Servers: 2

At time 0 3 customers arrive at the bank!
Server 0  is serving customer #1
        Waited time:  0 minutes.
        Transaction time: 2 minutes.

Server 1  is serving customer #2
        Waited time:  0 minutes.
        Transaction time: 1 minutes.

At time 1 3 customers arrive at the bank!
Server 1  is serving customer #3
        Waited time:  1 minutes.
        Transaction time: 1 minutes.
```

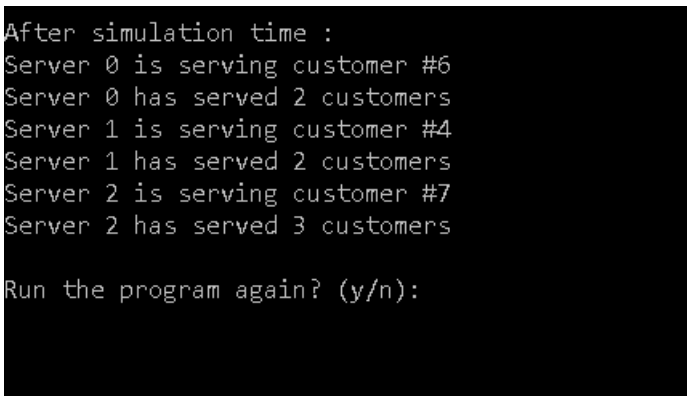
Here, server 1 serves the third customer instead of server 0 because the transaction time of the customer served by server 1(1 minute) is less than that by server 0(2 minutes).

Therefore, the implementation works correctly.

Screenshot:

Output Sample

```
#########################################################
**** BANK QUEUE SIMULATION PROGRAM ****
#########################################################

Please input the following data(Time in minutes):

Length of the Simulation(in minutes): 5
Average Number of Servers: 3

At time 0 4 customers arrive at the bank!
Server 0  is serving customer #1
        Waited time:  0 minutes.
        Transaction time: 2 minutes.

Server 1  is serving customer #2
        Waited time:  0 minutes.
        Transaction time: 1 minutes.

Server 2  is serving customer #3
        Waited time:  0 minutes.
        Transaction time: 1 minutes.

At time 1 4 customers arrive at the bank!
Server 1  is serving customer #4
        Waited time:  1 minutes.
        Transaction time: 4 minutes.

Server 2  is serving customer #5
        Waited time:  0 minutes.
        Transaction time: 3 minutes.

At time 2 2 customers arrive at the bank!
Server 0  is serving customer #6
        Waited time:  1 minutes.
        Transaction time: 4 minutes.

At time 3 1 customers arrive at the bank!
At time 4 1 customers arrive at the bank!
Server 2  is serving customer #7
        Waited time:  3 minutes.
        Transaction time: 1 minutes.


#################
* Data Output *
#################
Simulation Time:                      5 minutes
Average Transaction Time:             2.28571 minutes
Average Number of Servers:            3 servers
Average Total Wait Time:              1.50 minutes
Customers in line at end of simulation: 5 customers
```


After simulation time :
Server 0 is serving customer #6
Server 0 has served 2 customers
Server 1 is serving customer #4
Server 1 has served 2 customers
Server 2 is serving customer #7
Server 2 has served 3 customers
Run the program again? (y/n):