Laboratory 4

Queue ADT

I. OBJECTIVES:

In this laboratory, you will

- apply the concepts of a Queue ADT
- create array-based implementation of Linear and Circular Queue ADT
- and develop algorithm that will utilize the different operations of queue.

II. DISCUSSION:

A queue is an ordered collection of items from which items may be deleted at one end (called the *front* of a queue) and into which items maybe inserted at the other end (called the *rear* of the queue). The figure below illustrates a queue containing 3 elements A, B and C. A is at the front of the queue and C is at the rear. In Figure 4.1.2, an element has been deleted from the queue. Since elements may be deleted only from the front queue, A is removed and B now at the front. In Figure 4.1.3, when items D and E are inserted, they must be inserted at the rear of the queue.

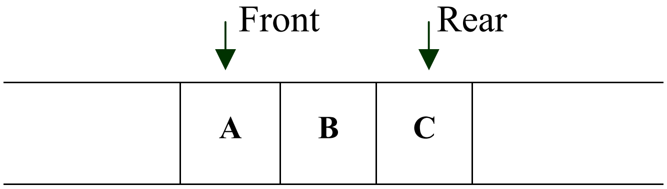

Fig 4.1.1 Queue

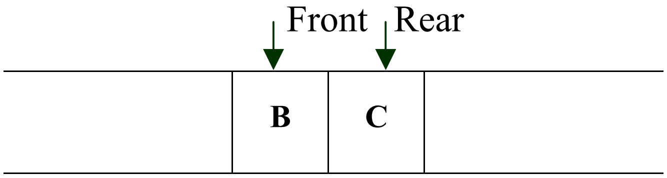

Fig 4.1.2 Queue

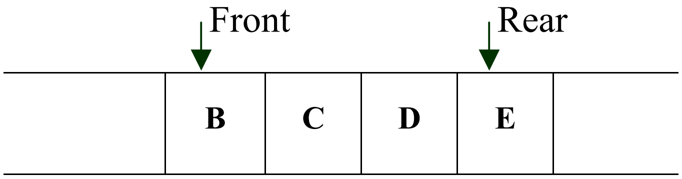

Fig 4.1.3 Queue

Queue Operations

Insert (q, x)

`Insert item` **`x`** `at the rear of the queue` **`q.`**

X=Remove(q)

```
Deletes the front element from the queue q and sets X to its
contents
```

Empty (q)

```
Returns false or true, it depends whether or not the queue
contains any element.
```

Underflow and Overflow

The insert operation can always be performed, since there is no limit to the number of elements a queue may contain. The queue will never encounter *overflow*. The remove operation, however, can be applied only if the queue is nonempty; there is no way to remove an element from a queue containing no elements. The result of an illegal attempt to remove an element from an empty queue is called *underflow*

III. Test and Debug

1. Create a test program that will implement the linear queue ADT's operations using array.

   Insert (q, x)
   X=Remove(q)
   Empty (q)

   Notes:
   1. *Consider that the element type (eltype) of the queue as character and maximum length is 5.*
   2. *The library of QUEUE functions is available in c:\datsala\lab4*

2. Test each queue operation using the following commands:

Commands	Results/Outputs
-e	Empty(q)
- + A	Insert(q,x)
-	x = remove(q)

3. Do the following sample runs

Commands	Output	
-e	True	
-+ A	A	
-+ B	A B	
-+ C	A B C	
--	B C	x = A
-+ D	B C D	
-+ E	B C D E	
-+ D	Overflow	
- - (perform 5x)	Underflow	

IV. SUPPLEMENTARY EXERCISES:

Consider circular queue applying the convention that the value of queue front is the array index immediately preceding the first element of the queue rather than the index of the first element itself.

1. Modify the linear queue program created in test & debug to simulate a circular queue with convention.
2. and do the following test runs to verify the algorithm:

Commands	Output	
-e	True	
-+ A	A	
-+ B	A B	
-+ C	A B C	
--	B C	x = A
-+ D	B C D	
-+ E	B C D E	
--	C D E	
-+ F	F C D E	
-+ G	Overflow	
- - (perform 5x)	Underflow	

V. Machine Problem

Simulate the flow of customers through the line during a time period **n** minutes long using the following algorithm

```
Initialize the queue to empty.

for (minute = 0; minute <n ; ++minute)
  {
        If the queue is not empty, then remove the customer at the front of the queue.
        Compute a random number k between 0 and 3.
        If k is 1, then add one customer to the line. If k is 2, then add 2 customers to the
        line. Otherwise (if k is 0 or 3), do not add any customers to the line.
  }
```

Calling the *rand()* function is a simple way to generate psuedo-random numbers. It should be available through the <stdlib> function set.

Your program should update the following information during each simulated minute; that is, during each pass through the loop:

- The total number of customers served
- The combined length of time these customers spent waiting in line (Average Wait)
- The maximum length of time any of these customers spent in line (Longest wait)

To compute how long a customer waited to be served, you need to store the "minute" that the customer was added to the queue as part of the queue data item corresponding to the customer.

Use your program to simulate the flow **of customers through the line** and **complete the table shown in Table 4.1**. Note that the average wait is the combined waiting time divided by the total number of customers served. Refer to the given example.

Table 4.1: Summary of Simulations

Time	Total Numbers of Customers Served	Average Wait	Longest Wait
30			
60			
120			
480			

Example of Simulations:

Initial:

K=1

C1:0		

K=2

C2:0	C3:0	

K=1

C3:10	C4:0	

K=0

C4:10		

Example: Summary of Simulation n(time)=30

Time	**Total Numbers of Customers Served**	**Average Wait**	**Longest Wait**
30	4	20/4=5	10

Note: *The instructor may change the machine problem.*

Queue ADT

DATA & RESULTS SHEET

(Tentative Laboratory Report)

Name: ____

Schedule:______________________ Section:______________

Date: _______________________ Grade: ______________

Test & Debug:

<table>
<tr><th>Step No.</th><th>Answers/Results</th></tr>
<tr><td>1&2</td><td>Attach source codes</td></tr>
<tr><td>3</td><td>
<table>
<tr><th>Commands</th><th>Output</th><th></th></tr>
<tr><td>-e</td><td>True</td><td></td></tr>
<tr><td>-+ A</td><td>A</td><td></td></tr>
<tr><td>-+ B</td><td>A B</td><td></td></tr>
<tr><td>-+ C</td><td>A B C</td><td></td></tr>
<tr><td>--</td><td>B C</td><td>x = A</td></tr>
<tr><td>-+ D</td><td>B C D</td><td></td></tr>
<tr><td>-+ E</td><td>B C D E</td><td></td></tr>
<tr><td>-+ D</td><td>Overflow</td><td></td></tr>
<tr><td>- - (perform 5x)</td><td>Underflow</td><td></td></tr>
</table>
</td></tr>
</table>

Supplementary Exercises:

<table>
<tr><th>Supp No.</th><th>Answers/Results</th></tr>
<tr><td>1</td><td>Attach source codes</td></tr>
<tr><td>2</td><td>
<table>
<tr><th>Commands</th><th>Output</th><th></th></tr>
<tr><td>-e</td><td>True</td><td></td></tr>
<tr><td>-+ A</td><td>A</td><td></td></tr>
<tr><td>-+ B</td><td>A B</td><td></td></tr>
<tr><td>-+ C</td><td>A B C</td><td></td></tr>
<tr><td>--</td><td>B C</td><td>x = A</td></tr>
<tr><td>-+ D</td><td>B C D</td><td></td></tr>
<tr><td>-+ E</td><td>B C D E</td><td></td></tr>
<tr><td>--</td><td>C D E</td><td></td></tr>
<tr><td>-+ F</td><td>F C D E</td><td></td></tr>
<tr><td>-+ G</td><td>Overflow</td><td></td></tr>
<tr><td>- - (perform 5x)</td><td>Underflow</td><td></td></tr>
</table>
</td></tr>
</table>

Machine Problem:

Requirements	Completed?		Remarks
	YES	**NO**	
Simulation			
Table			

Note: Check the column YES if completed otherwise check the column NO.(for instructors only)

INSTRUCTOR'S SIGNATURE: ______________________________