EENG212
Lab 5
Queues

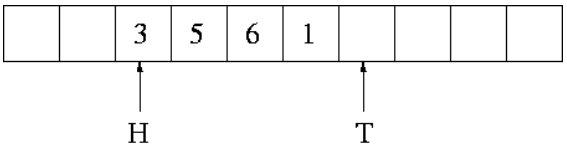
A queue is an ordered collection of items from which items may be deleted at one end (called the front of the queue) and into which items may be inserted at the other end (called the rear of the queue). The first element inserted into the queue is the first element to be removed. For this reason a queue is sometimes called a ***fifo*** (first-in first-out) list as opposed to the stack, which is a ***lifo*** (last-in first-out).
A queue may be implemented in C in the following way:

```
#define MAXQUEUE 10
struct queue{
        int items[MAXQUEUE];
        int front,rear;
};
```

Circular Queue:

A circular queue is a particular implementation of a queue. It is very efficient. It is also quite useful in low level code, because insertion and deletion are totally independent, which means that you don't have to worry about an interrupt handler trying to do an insertion at the same time as your main code is doing a deletion.

A circular queue consists of an array that contains the items in the queue, two array indexes and an optional length. The indexes are called the *head* and *tail* pointers and are labelled H and T on the diagram.



The head pointer points to the first element in the queue, and the tail pointer points just beyond the last element in the queue. If the tail pointer is before the head pointer, the queue wraps around the end of the array.

Is the queue empty or full? There is a problem with this: Both an empty queue and a full queue would be indicated by having the head and tail point to the same element. There are two ways around this: either maintain a variable with the number of items in the queue, or create the array with one more element that you will actually need so that the queue is never full.

The following program implements a circular queue using the queue structure given above.

```
#include <stdio.h>
#include<stdlib.h>
#define FALSE 0
#define TRUE 1
#define MAXQUEUE 10
struct queue {
   int  front,rear;
   int items[MAXQUEUE];
};
void Menu (int *choice);
int empty(queue*);
int remove(queue*);
void insert(queue* , int );v
oid PrintQueue (queue*); i
nt isdigit(char );
void main () {
    queue q;
    int data,choice,y;
    q.front=q.rear=MAXQUEUE-1;
   do {
    Menu (&choice);
    switch (choice) {
        case 1:
              printf ("Enter data item value to add  ");
              scanf ("%d", &data);
              insert(&q,data);
          break;
        case 2:
              y= remove(&q);
            printf("Element removed is %d",y);
           break;
        case 3:
              PrintQueue (&q);
          break;
        case 4:
              exit(1);
      default:
         printf ("Invalid menu choice - try again\n");
         break;}
   } while (choice != 4);
}

void Menu (int *choice)
{
  char    local;
   printf("----------------------------------------");
   printf ("\nEnter\t1 to add item,\n\t2 to remove item\n\t3 to print queue\n\t4
to quit\n");
```

```
    do {
        local = getchar ();
        if ((isdigit (local) == FALSE) && (local != '\n')) {
            printf ("\nyou must enter an integer.\n");
            printf ("Enter 1 to add, 2 to remove, 3 to print, 4 to quit\n");
        }
   }while (isdigit ((unsigned char) local) == FALSE);
   *choice = (int) local - '0';
}

int empty(queue *pq)
{
    if(pq->front==pq->rear)
        return(TRUE);
    return(FALSE);
}

void insert(queue *pq, int x)
{
    /* make room for new element*/
    if(pq->rear==MAXQUEUE-1)
        pq->rear=0;
    else
        (pq->rear)++;
    /* check for overflow*/
    if(pq->rear==pq->front)
    {
        printf("Queue overflow!");
        exit(1);
    }
    pq->items[pq->rear]=x;
    return;
}

int remove(queue *pq){
    if(empty(pq))
    {
        printf("Queue is empty");
          return 0;
    }
    if(pq->front == MAXQUEUE-1)
        pq->front=0;
    else
        (pq->front)++;
    return(pq->items[pq->front]);
}

void PrintQueue (queue *pq)
 {
    int x,y;
```

```
    if (empty(pq))
        printf ("Queue is empty!\n");
    else{
        x=pq->rear;
        y=pq->front;
        if(y==MAXQUEUE-1)
                y=-1;
        printf ("\n");
        while (x>y)
           {
                printf ("%d\t", pq->items[x]);
                x--;
        }
        printf ("\n");
      }
}
int isdigit(char symb)
{
    return(symb>='0' && symb <='9');
}
```

HW 6(Due after one week)

Write a program and fill a queue with random numbers between 0 and 100. The size of the queue is assumed to be 15. After filling the array with random numbers, display the elements in the queue and remove the elements of the queue and store these numbers according to the following criteria.

If the number in the queue is less than 50, remove it from the queue and store it inside queue 2.
Else, remove the number and store these values inside queue 3.
Display all three queues on the screen.

NOTES:

1) Please Send your Homework in the following Emails, but remember ***who is your lab instructor***

emu.clab2@gmail.com *for* ***Pouya's*** *Student*
eenglab212@gmail.com *for* ***Mohamad****'s Student*

2) Subject of email Should include student Number + Homework Number
For Example: "St. 15000012 Homework #1"

3) Your homework should be saved with your student number and attached as notepad.
For Example 15000012.txt