Name : Deshmukh Mehmood Rehan

MIS No. : 612303050

SY Computer Science – Div 1

Question: *Implement a queue of structures. The queue stores the following structure:*

typedef struct data {

char name[16];

unsigned int age;

}data;

Write the following queue functions: qinit, enq, deq, qfull, qempty;

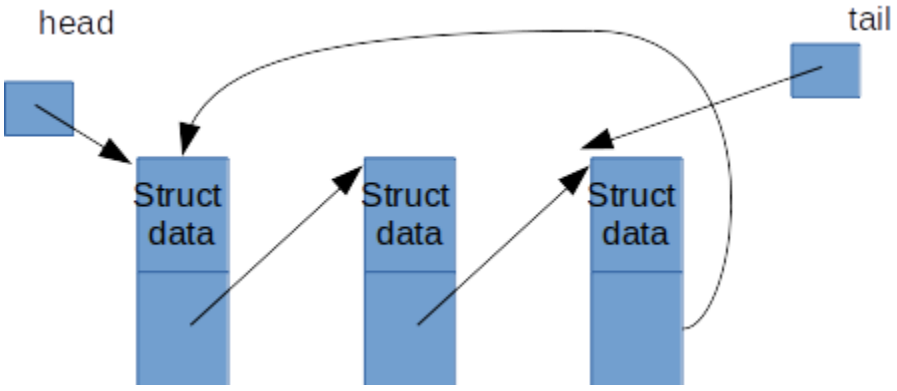
Define the proper prototypes in queue.h and write the code in queue.c

The queue implementation should be done using singly linked circularly connected structures (nodes), where each node will store one instance of 'data'.

You can use head and tail - both pointers

In qfull, you may simply assume that the queue is never full and write code accordingly. The real challenging part (for no extra marks) is to detect full if malloc fails, but still make sure that qfull() can detect it and enq() always works if qfull() said not-full.

Here is a diagram of how your queue will store the data: Note that tail is optional and you may or may not use it.



Do a proper typedef of the queue type in queue.h

Use this main() function in a file called testqueue.c to test your queue. Make sure that your code compiles with this code and you can test it.

```
#include <stdio.h>

#include "queue.h"
int main() {
  queue q;
  data d;

  qinit(&q);
  while(scanf("%s%u", d.name, &(d.age)) != -1)
    if(!qfull(&q))
      enq(&q, d);
  while(!qempty(&q)) {
    d = deq(&q);
    printf("%s %u\n", d.name, d.age);
  }
  return 0;
}
```

Code:

header.h : This File includes the declarations of structures and function prototypes

```
/*

Write files: queue.c  queue.h and testqueue.c
Submit a single file: MISID.tar.gz
Implement a queue of structures. The queue stores the following structure:
typedef struct data {
    char name[16];
    unsigned int age;
}data;

Write the following queue functions: qinit, enq, deq, qfull, qempty;
Define the proper prototypes in queue.h and write the code in queue.c
The queue implementation should be done using singly linked circularly
connected structures (nodes), where each node will store one instance of
'data'.
You can use head and tail - both pointers
In qfull, you may simply assume that the queue is never full and write code
accordingly. The real challenging part (for no extra marks) is to detect full
```

```
if malloc fails, but still make sure that qfull() can detect it and enq()
always works if qfull() said not-full.
Here is a diagram of how your queue will store the data: Note that tail is
optional and you may or may not use it.
Do a proper typedef of the queue type in queue.h
Use this main() function in a file called  testqueue.c   to test your queue.
Make sure that your code compiles with this code and you can test it.

#include <stdio.h>

#include "queue.h"
int main() {
    queue q;
    data d;

    qinit(&q);
    while(scanf("%s%u", d.name, &(d.age)) != -1)
        if(!qfull(&q))
            enq(&q, d);
    while(!qempty(&q)) {
        d = deq(&q);
        printf("%s %u\n", d.name, d.age);
    }
    return 0;
}

*/
#include <stdlib.h>
#include <stdio.h>

/*
This is the structure of Data that will be stored in the queue
It consists of:
    - name: a string of 16 characters
    - age: an unsigned integer
*/
typedef struct Data {
    char name[16];
    unsigned int age;
} Data;

/*
This is the structure of a Node that will be used to store the data in the
queue
It consists of:
    - data: The Data structure mentioned above
    - next: a pointer to the next Node
*/
```

```c
typedef struct Node {
    Data data;
    struct Node *next;
} Node;

/*
This is the structure of the Queue
It consists of:
    - head: a pointer to the first Node in the queue
    - tail: a pointer to the last Node in the queue
*/
typedef struct Queue {
    Node *head;
    Node *tail;
} Queue;

/* Function prototypes */
void qinit(Queue *q);
void enq(Queue *q, Data data);
Data deq(Queue *q);
int qempty(Queue *q);
```

logic.c : contains the definition of all the functions declared in the header file along with some helper functions

```c
#include "header.h"

/*
This function initializes the queue
It takes a pointer to the queue as an argument
It sets the head and tail of the queue to NULL
*/
void qinit(Queue *q) {
    q->head = NULL;
    q->tail = NULL;
}
/*
This function checks if the queue is empty
It takes a pointer to the queue as an argument
It returns 1 if the queue is empty and 0 otherwise
The queue is empty if the head is NULL
*/
```

```c
int qempty(Queue *q) {
    return q->head == NULL;
}

/*
This function adds an element to the queue
It takes a pointer to the queue and the data to be added as arguments
It creates a new node with the data and adds it to the end of the queue

If the queue is empty, the new node is both the head and the tail of the queue
Otherwise, the new node is added after the tail and becomes the new tail
The new tail points to the head of the queue to maintain the circular
structure
*/

void enq(Queue *q, Data data){
    Node *newNode = (Node *)malloc(sizeof(Node));
    if(!newNode) return;
    newNode->data = data;
    newNode->next = NULL;

    if(qempty(q)) {
        q->head = newNode;
        q->tail = newNode;
    } else {
        q->tail->next = newNode;
        q->tail = newNode;
        q->tail->next = q->head;
    }
}

/*
This function removes an element from the queue
It takes a pointer to the queue as an argument
It returns the data of the removed element

If the queue is empty, it returns an empty data structure
Otherwise, it removes the head of the queue and returns its data
If the queue becomes empty after removing the element, it sets the head and
tail to NULL
Otherwise, it updates the head of the queue to the next element
The tail of the queue points to the new head to maintain the circular
structure
*/

Data deq(Queue *q){
    if(qempty(q)) {
        Data data = {"", -1};
```

```
        return data;
    }

    Node *removedNode = q->head;
    Data removedData = removedNode->data;

    if(q->head == q->tail) {
        q->head = q->tail = NULL;
    } else {
        q->head = q->head->next;
        q->tail->next = q->head;
    }

    free(removedNode);
    return removedData;
}
```

main.c : This contains the code to test the implementation

```
/* This is the test file provided in the question */


#include "header.h"

int main() {
    Queue q;
    Data d;

    qinit(&q);
    while(scanf("%s%u", d.name, &(d.age)) != -1) enq(&q, d);

    while(!qempty(&q)) {
        d = deq(&q);
        printf("Name: %s, Age: %u\n", d.name, d.age); /* Modified the output
format */
    }
    return 0;
}
```

Output:

```
Assignment 4>gcc .\main.c .\logic.c -Wall -o .\main
Assignment 4>.\main
Mehmood 19
Yashwant 19
Deven 21
Name: Mehmood, Age: 19
Name: Yashwant, Age: 19
Name: Deven, Age: 21
Assignment 4>
```