Name : Deshmukh Mehmood Rehan

MIS No. : 612303050

SY Computer Science – Div 1

Question: *Implement a queue of integers using an ADT list. Invoke all queue operations using a menu-driven program.*

Code:

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

```c
/* Implement a queue of integers using an ADT list. Invoke all queue
operations using a menu-driven program. */

#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
/*
This is the structure of a node in the queue.
It contains:
    - data: the integer value of the node
    - next: a pointer to the next node in the queue
*/

typedef struct Node{
    int data;
    struct Node *next;
} Node;

/*
This is the structure of the queue.
It contains:
    - 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;

/*These are the function prototypes for the queue operations*/
void init(Queue *q);
void enqueue(Queue *q, int data);
int dequeue(Queue *q);
int peek(Queue *q);
```

```
int isEmpty(Queue *q);
void display(Queue *q);
void clear(Queue *q);
```

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

```
#include "header.h"

/*
This function initializes the queue.
It sets the head and tail pointers to NULL.
*/

void init(Queue *q){
    q->head = NULL;
    q->tail = NULL;
}

/*
This function adds a new node to the queue.
It takes the queue and the data to be added as arguments.

It creates a new node and sets its data to the given data.
If the queue is empty, it sets the head and tail pointers to the new node.
Otherwise, it adds the new node to the end of the queue and updates the tail
pointer.
*/

void enqueue(Queue *q, int data){
    Node *newNode = (Node *)malloc(sizeof(Node));
    if(!newNode) return;

    newNode->data = data;
    newNode->next = NULL;

    if(q->head == NULL){
        q->head = newNode;
        q->tail = newNode;
    }else{
        q->tail->next = newNode;
        q->tail = newNode;
    }
}
/*
This function removes a node from the queue.
```

```c
It takes the queue as an argument and returns the data of the removed node.

If the queue is empty, it returns INT_MIN.
Otherwise, it removes the first node from the queue and returns its data.
It also updates the head pointer to point to the next node in the queue.

if the node removed is the only node in the queue, it also updates the tail
pointer.
*/

int dequeue(Queue *q){
    if(isEmpty(q)) return INT_MIN;

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

    q->head = q->head->next;
    free(removedNode);

    return removedData;
}

/*
This function returns the data of the first node in the queue.
It takes the queue as an argument.

If the queue is empty, it returns INT_MIN.
*/

int peek(Queue *q){
    if(isEmpty(q)) return INT_MIN;

    return q->head->data;
}

/*
This function checks if the queue is empty.
*/
int isEmpty(Queue *q){
    return q->head == NULL;
}

/*
This function displays the elements of the queue.
*/

void display(Queue *q){
    if(isEmpty(q)) return;
```

```
    Node *temp = q->head;
    while(temp){
        printf("%d | ", temp->data);
        temp = temp->next;
    }
    printf("\b\b  \n");
}

/*
This function clears the queue.
*/

void clear(Queue *q){
    while(!isEmpty(q)){
        dequeue(q);
    }
}
```

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

```
#include "header.h"

void printMenu();
void handleChoice(Queue *q);


int main(){
    Queue q;
    init(&q);
    while(1){
        printMenu();
        handleChoice(&q);
    }
    return 0;
}

void printMenu(){
    printf("\n\n**********Array ADT Menu:**********\n\n");
    printf("Select any of the instructions given below:\n");
    printf("Enter your choice:\n");
    printf("1. Enqueue an element in the queue.\n");
    printf("2. Dequeue an element from the queue.\n");
    printf("3. Peek the front element of the queue.\n");
    printf("4. Check if the queue is empty.\n");
    printf("5. Display the queue.\n");
    printf("6. Clear the queue.\n");
    printf("7. Exit\n");
```

```c
}

void handleChoice(Queue *q){
    int choice, data;
    printf("Enter your choice: ");
    scanf("%d", &choice);
    switch(choice){
        case 1:
            printf("Enter the data to enqueue: ");
            scanf("%d", &data);
            enqueue(q, data);
            break;
        case 2:
            data = dequeue(q);
            if(data == INT_MIN){
                printf("Queue is empty!\n");
            }else{
                printf("Dequeued data: %d\n", data);
            }
            break;
        case 3:
            data = peek(q);
            if(data == INT_MIN){
                printf("Queue is empty!\n");
            }else{
                printf("Peeked data: %d\n", data);
            }
            break;
        case 4:
            if(isEmpty(q)){
                printf("Queue is empty!\n");
            }else{
                printf("Queue is not empty!\n");
            }
            break;
        case 5:
            display(q);
            break;
        case 6:
            clear(q);
            printf("Queue cleared!\n");
            break;
        case 7:
            exit(0);
        default:
            printf("Invalid choice!\n");
    }
}
```

Output:

1. **Enqueue**

```
Labwork 13 Queue>gcc .\logic.c .\main.c -Wall -o .\main
Labwork 13 Queue>.\main


**********Array ADT Menu:**********

Select any of the instructions given below:
Enter your choice:
1. Enqueue an element in the queue.
2. Dequeue an element from the queue.
3. Peek the front element of the queue.
4. Check if the queue is empty.
5. Display the queue.
6. Clear the queue.
7. Exit
Enter your choice: 1
Enter the data to enqueue: 12


**********Array ADT Menu:**********

Select any of the instructions given below:
Enter your choice:
1. Enqueue an element in the queue.
2. Dequeue an element from the queue.
3. Peek the front element of the queue.
4. Check if the queue is empty.
5. Display the queue.
6. Clear the queue.
7. Exit
Enter your choice: 1
Enter the data to enqueue: 23
```

2. **Display**

```
**********Array ADT Menu:**********

Select any of the instructions given below:
Enter your choice:
1. Enqueue an element in the queue.
2. Dequeue an element from the queue.
3. Peek the front element of the queue.
4. Check if the queue is empty.
5. Display the queue.
6. Clear the queue.
7. Exit
Enter your choice: 5
12 | 23
```

3. **Peek**

```
**********Array ADT Menu:**********

Select any of the instructions given below:
Enter your choice:
1. Enqueue an element in the queue.
2. Dequeue an element from the queue.
3. Peek the front element of the queue.
4. Check if the queue is empty.
5. Display the queue.
6. Clear the queue.
7. Exit
Enter your choice: 3
Peeked data: 12
```

4. **Dequeue**

```
**********Array ADT Menu:**********

Select any of the instructions given below:
Enter your choice:
1. Enqueue an element in the queue.
2. Dequeue an element from the queue.
3. Peek the front element of the queue.
4. Check if the queue is empty.
5. Display the queue.
6. Clear the queue.
7. Exit
Enter your choice: 2
Dequeued data: 12
```

5. **Clear**

```
**********Array ADT Menu:**********

Select any of the instructions given below:
Enter your choice:
1. Enqueue an element in the queue.
2. Dequeue an element from the queue.
3. Peek the front element of the queue.
4. Check if the queue is empty.
5. Display the queue.
6. Clear the queue.
7. Exit
Enter your choice: 6
Queue cleared!
```

6. **isEmpty**

```
**********Array ADT Menu:**********

Select any of the instructions given below:
Enter your choice:
1. Enqueue an element in the queue.
2. Dequeue an element from the queue.
3. Peek the front element of the queue.
4. Check if the queue is empty.
5. Display the queue.
6. Clear the queue.
7. Exit
Enter your choice: 4
Queue is empty!
```