QUEUES

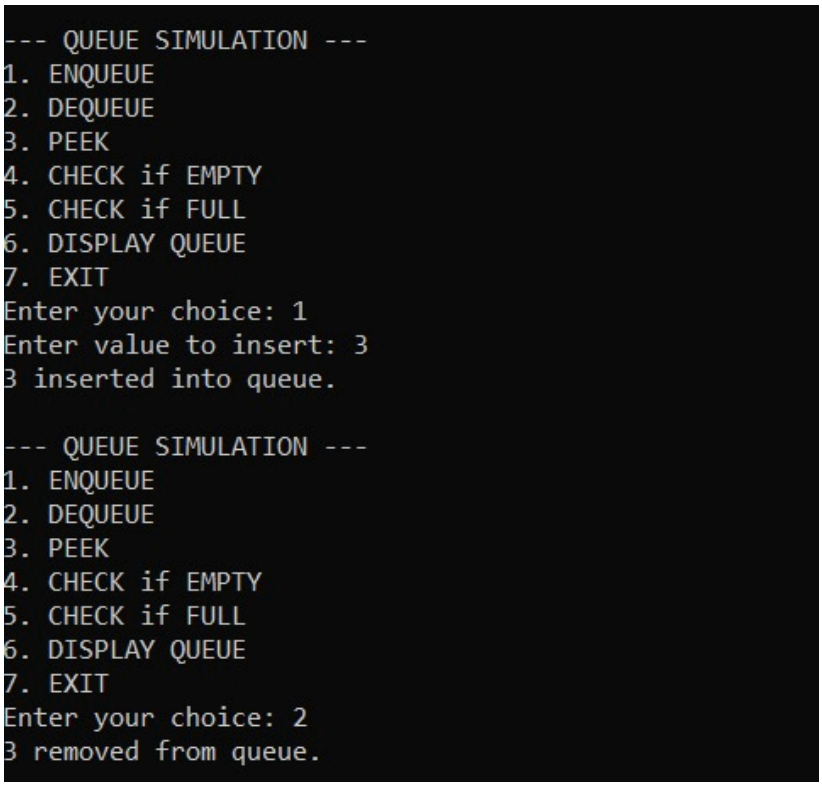
```
#include <stdio.h>
#include <stdlib.h>

#define MAX 100

int queue[MAX];
int front = -1;
int rear = -1;

// Check if queue is full
int isFull() {
    return rear == MAX - 1;
}

// Check if queue is empty
int isEmpty() {
    return front == -1 || front > rear;
}

// ENQUEUE operation
void enqueue(int x) {
    if (isFull()) {
        printf("Queue Overflow! Cannot insert %d\n", x);
        return;
    }

    if (front == -1)
        front = 0;

    rear++;
    queue[rear] = x;
    printf("%d inserted into queue.\n", x);
}

// DEQUEUE operation
int dequeue() {
    if (isEmpty()) {
        printf("Queue Underflow! Queue is empty.\n");
        return -1;
    }

    int x = queue[front];
```

```
    front++;

    if (front > rear) {   // Reset after last deletion
        front = -1;
        rear = -1;
    }

    printf("%d removed from queue.\n", x);
    return x;
}

// PEEK operation
int peek() {
    if (isEmpty()) {
        printf("Queue is empty! No front element.\n");
        return -1;
    }

    printf("Front element is: %d\n", queue[front]);
    return queue[front];
}

// DISPLAY queue elements
void display() {
    if (isEmpty()) {
        printf("Queue is empty!\n");
        return;
    }

    printf("Queue elements (front to rear): ");
    for (int i = front; i <= rear; i++) {
        printf("%d ", queue[i]);
    }
    printf("\n");
}

// MAIN MENU
int main() {
    int choice, value;

    while (1) {
        printf("\n--- QUEUE SIMULATION ---\n");
        printf("1. ENQUEUE\n");
        printf("2. DEQUEUE\n");
```

```
printf("3. PEEK\n");
printf("4. CHECK if EMPTY\n");
printf("5. CHECK if FULL\n");
printf("6. DISPLAY QUEUE\n");
printf("7. EXIT\n");

printf("Enter your choice: ");
scanf("%d", &choice);

switch (choice) {
    case 1:
        printf("Enter value to insert: ");
        scanf("%d", &value);
        enqueue(value);
        break;

    case 2:
        dequeue();
        break;

    case 3:
        peek();
        break;

    case 4:
        if (isEmpty())
            printf("Queue is EMPTY.\n");
        else
            printf("Queue is NOT empty.\n");
        break;

    case 5:
        if (isFull())
            printf("Queue is FULL.\n");
        else
            printf("Queue is NOT full.\n");
        break;

    case 6:
        display();
        break;

    case 7:
        printf("Exiting...\n");
```

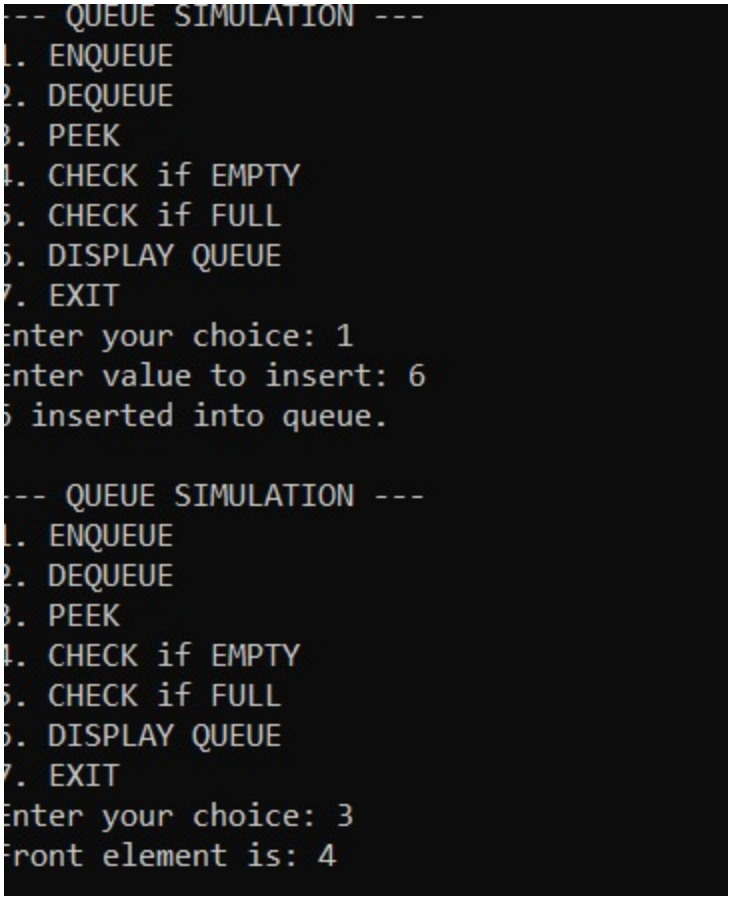
```
            exit(0);

        default:
            printf("Invalid Choice! Try again.\n");
      }
   }

   return 0;
}
```

OUTPUTS



```
--- QUEUE SIMULATION ---
1. ENQUEUE
2. DEQUEUE
3. PEEK
4. CHECK if EMPTY
5. CHECK if FULL
6. DISPLAY QUEUE
7. EXIT
Enter your choice: 4
Queue is EMPTY.

--- QUEUE SIMULATION ---
1. ENQUEUE
2. DEQUEUE
3. PEEK
4. CHECK if EMPTY
5. CHECK if FULL
6. DISPLAY QUEUE
7. EXIT
Enter your choice: 6
Queue is empty!
```

```
--- QUEUE SIMULATION ---
1. ENQUEUE
2. DEQUEUE
3. PEEK
4. CHECK if EMPTY
5. CHECK if FULL
6. DISPLAY QUEUE
7. EXIT
Enter your choice: 1
Enter value to insert: 4
4 inserted into queue.

--- QUEUE SIMULATION ---
1. ENQUEUE
2. DEQUEUE
3. PEEK
4. CHECK if EMPTY
5. CHECK if FULL
6. DISPLAY QUEUE
7. EXIT
```


--- QUEUE SIMULATION ---
1. ENQUEUE
2. DEQUEUE
3. PEEK
4. CHECK if EMPTY
5. CHECK if FULL
6. DISPLAY QUEUE
7. EXIT
Enter your choice: 1
Enter value to insert: 6
6 inserted into queue.
--- QUEUE SIMULATION ---
1. ENQUEUE
2. DEQUEUE
3. PEEK
4. CHECK if EMPTY
5. CHECK if FULL
6. DISPLAY QUEUE
7. EXIT
Enter your choice: 3
Front element is: 4