

Rajalakshmi Engineering College

Name: sugitha nesamani
Email: 241801279@rajalakshmi.edu.in
Roll no: 241801279
Phone: 8637611457
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 4_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

In a bustling IT department, staff regularly submit helpdesk tickets to request technical assistance. Managing these tickets efficiently is vital for providing quality support.

Your task is to develop a program that uses an array-based queue to handle and prioritize helpdesk tickets based on their unique IDs.

Implement a program that provides the following functionalities:

Enqueue Helpdesk Ticket: Add a new helpdesk ticket to the end of the queue. Provide a positive integer representing the ticket ID for the new ticket. Dequeue Helpdesk Ticket: Remove and process the next helpdesk ticket from the front of the queue. The program will display the ticket ID of the processed ticket. Display Queue: Display the ticket IDs of all the

helpdesk tickets currently in the queue.

Input Format

The input consists of integers corresponding to the operation that needs to be performed:

Choice 1: Enqueue the ticket ID into the queue. If the choice is 1, the following input is a space-separated integer, representing the ticket ID to be enqueued into the queue.

Choice 2: Dequeue a ticket from the queue.

Choice 3: Display the ticket IDs in the queue.

Choice 4: Exit the program.

Output Format

The output displays messages according to the choice and the status of the queue:

If the choice is 1:

1. Insert the given ticket ID into the queue and display "Helpdesk Ticket ID [id] is enqueued." where [id] is the ticket ID that is inserted.
2. If the queue is full, print "Queue is full. Cannot enqueue."

If the choice is 2:

1. Dequeue a ticket ID from the queue and display "Dequeued Helpdesk Ticket ID: " followed by the corresponding ID that is dequeued.
2. If the queue is empty without any elements, print "Queue is empty."

If the choice is 3:

1. The output prints "Helpdesk Ticket IDs in the queue are: " followed by the space-separated ticket IDs present in the queue.
2. If there are no elements in the queue, print "Queue is empty."

If the choice is 4:

1. Exit the program and print "Exiting the program"

If any other choice is entered, print "Invalid option."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1 101

1 202

1 203

1 204

1 205

1 206

3

2

3

4

Output: Helpdesk Ticket ID 101 is enqueued.

Helpdesk Ticket ID 202 is enqueued.

Helpdesk Ticket ID 203 is enqueued.

Helpdesk Ticket ID 204 is enqueued.

Helpdesk Ticket ID 205 is enqueued.

Queue is full. Cannot enqueue.

Helpdesk Ticket IDs in the queue are: 101 202 203 204 205

Dequeued Helpdesk Ticket ID: 101

Helpdesk Ticket IDs in the queue are: 202 203 204 205

Exiting the program

Answer

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

```
#define MAX_SIZE 5
```

```
int ticketIDs[MAX_SIZE];
```

```
int front = -1;
```

```
int rear = -1;
```

```
int lastDequeued;
```

```
void initializeQueue() {
```

```
    front = -1;
```

```
    rear = -1;
```

```
}
```

```

int isEmpty() {
    return front == -1; // Queue is empty if front is -1
}

// Function to check if the queue is full
int isFull() {
    return rear == MAX_SIZE - 1; // Queue is full if rear reaches the max size
}

// Function to enqueue a ticket ID
int enqueue(int ticketID) {
    if (isFull()) {
        printf("Queue is full. Cannot enqueue.\n");
        return 0; // Return 0 to indicate failure
    } else {
        if (front == -1) { // If it's the first element to be inserted
            front = 0;
        }
        rear++;
        ticketIDs[rear] = ticketID;
        printf("Helpdesk Ticket ID %d is enqueued.\n", ticketID);
        return 1; // Return 1 to indicate success
    }
}

// Function to dequeue a ticket ID
int dequeue() {
    if (isEmpty()) {
        return 0; // Return 0 to indicate failure (queue is empty)
    } else {
        lastDequeued = ticketIDs[front]; // Store the last dequeued ticket ID

        // If the queue becomes empty after dequeue
        if (front == rear) {
            front = rear = -1; // Reset the queue
        } else {
            front++; // Move the front pointer to the next ticket
        }

        return 1; // Return 1 to indicate success
    }
}

```

// Function to display the current tickets in the queue

```
void display() {  
    if (isEmpty()) {  
        printf("Queue is empty.\n");  
    } else {  
        printf("Helpdesk Ticket IDs in the queue are: ");  
        for (int i = front; i <= rear; i++) {  
            printf("%d ", ticketIDs[i]);  
        }  
        printf("\n");  
    }  
}
```

```
int main() {  
    int ticketID;  
    int option;  
    initializeQueue();  
    while (1) {  
        if (scanf("%d", &option) == EOF) {  
            break;  
        }  
        switch (option) {  
            case 1:  
                if (scanf("%d", &ticketID) == EOF) {  
                    break;  
                }  
                enqueue(ticketID);  
                break;  
            case 2:  
                if (dequeue()) {  
                    printf("Dequeued Helpdesk Ticket ID: %d\n", lastDequeued);  
                } else {  
                    printf("Queue is empty.\n");  
                }  
                break;  
            case 3:  
                display();  
                break;  
            case 4:  
                printf("Exiting the program\n");  
                return 0;  
            default:
```

```
        printf("Invalid option.\n");  
        break;  
    }  
    }  
    return 0;  
}
```

Status : Correct

Marks : 10/10