Vallurupalli Nageswara Rao Vignana Jyothi Institute of Engineering &Technology

Name :Dumpeta Varshitha . Email :varshithadumpeta@gmail.com
Roll no:23071A1023 Phone :8919552327
Branch :VNRVJIET Department :EIE
Batch :2027 Degree :B.Tech EIE

2023_27_II_EIE 1_Data Structures_lab

DATA STRUCTURES_CODING_WEEK 3

Attempt : 1
Total Mark : 20
Marks Obtained : 20

Section 1 : CODING

1. Problem Statement

Write a program for the implementation of the circular queue using an array.

Answer

```
// You are using GCC
#include <stdio.h>

#define MAX 20

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

void insert(int item) {
    if ((front == 0 && rear == MAX - 1) || (front == rear + 1)) {
        printf("Queue Overflow\n");
        return;
    }
```

```
    if (front == -1) {
        front = 0;
        rear = 0;
    } else {
        if (rear == MAX - 1)
rear = 0;
        else
            rear = rear + 1;
    }
    cqueue_arr[rear] = item;
}

void deleteElement() {
    if (front == -1) {
        printf("Queue Underflow\n");
        return;
    }

    printf("Element deleted from queue is: %d\n", cqueue_arr[front]);

    if (front == rear) {
        front = -1;
        rear = -1;
    } else {
        if (front == MAX - 1)
            front = 0;
        else
            front = front + 1;
    }
}

void display() {
    if (front == -1) {
        printf("Queue is empty\n");
        return;
    }

    int front_pos = front;
    int rear_pos = rear;
    printf("Queue elements:\n");
```

```
    if (front_pos <= rear_pos) {
        while (front_pos <= rear_pos) {
            printf("%d ", cqueue_arr[front_pos]);
            front_pos++;
        }
    } else {
        while (front_pos <= MAX - 1) {
            printf("%d ", cqueue_arr[front_pos]);
            front_pos++;
        }

        front_pos = 0;
        while (front_pos <= rear_pos) {
            printf("%d ", cqueue_arr[front_pos]);
            front_pos++;
        }
    }
    printf("\n");
}

int main() {
    int choice, item;
    do {
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                scanf("%d", &item);
                insert(item);
                break;
            case 2:
                deleteElement();
                break;
            case 3:
                display();
                break;
            case 4:
                return 0;
            default:
                printf("Wrong choice\n");
        }
    } while (1);
```

```
    return 0;
}
```

Status : Correct ***Marks : 10/10***

2. 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.

Answer

```
// You are using GCC
#include <stdio.h>
#define MAX_SIZE 5

int ticketIDs[MAX_SIZE];
int front = -1;
int rear = -1;

void initializeQueue() {
    front = -1;
    rear = -1;
}
```

```
int isEmpty() {
    return front == -1;
}

int isFull() {
    return (rear + 1) % MAX_SIZE == front;
}

int enqueue(int ticketID) {
    if (isFull()) {
        printf("Queue is full. Cannot enqueue.\n");
        return 0;
    }

    if (isEmpty()) {
        front = rear = 0;
    } else {
        rear = (rear + 1) % MAX_SIZE;
    }

    ticketIDs[rear] = ticketID;
    printf("Helpdesk Ticket ID %d is enqueued.\n", ticketID);
    return 1;
}

int dequeue(int* ticketID) {
    if (isEmpty()) {
        return 0;
    }

    *ticketID = ticketIDs[front];
    if (front == rear) {
        front = rear = -1;
    } else {
        front = (front + 1) % MAX_SIZE;
    }

    return 1;
}

void display() {
```

```
    if (isEmpty()) {
        printf("Queue is empty.\n");
    } else {
        printf("Helpdesk Ticket IDs in the queue are: ");
        int i = front;
        while (i != rear) {
            printf("%d ", ticketIDs[i]);
            i = (i + 1) % MAX_SIZE;
        }
        printf("%d\n", ticketIDs[rear]);
    }
}

int main() {
    int ticketID;
    int option;

    initializeQueue();

    while (1) {
        if (scanf("%d", &option) != 1) {
            break;
        }

        switch (option) {
            case 1:
                if (scanf("%d", &ticketID) != 1) {
                    break;
                }
                if (enqueue(ticketID)) {
                    // Helpdesk ticket enqueued successfully
                }
                break;

            case 2:
                if (dequeue(&ticketID)) {
                    printf("Dequeued Helpdesk Ticket ID: %d\n", ticketID);
                } else {
                    printf("Queue is empty.\n");
                }
                break;
```

```
            case 3:
                display();
                break;

            case 4:
                printf("Exiting the program");
                return 0;
                default:
                printf("Invalid option.\n");
                break;
}
}
return 0;
}
```

Status : Correct ***Marks : 10/10***