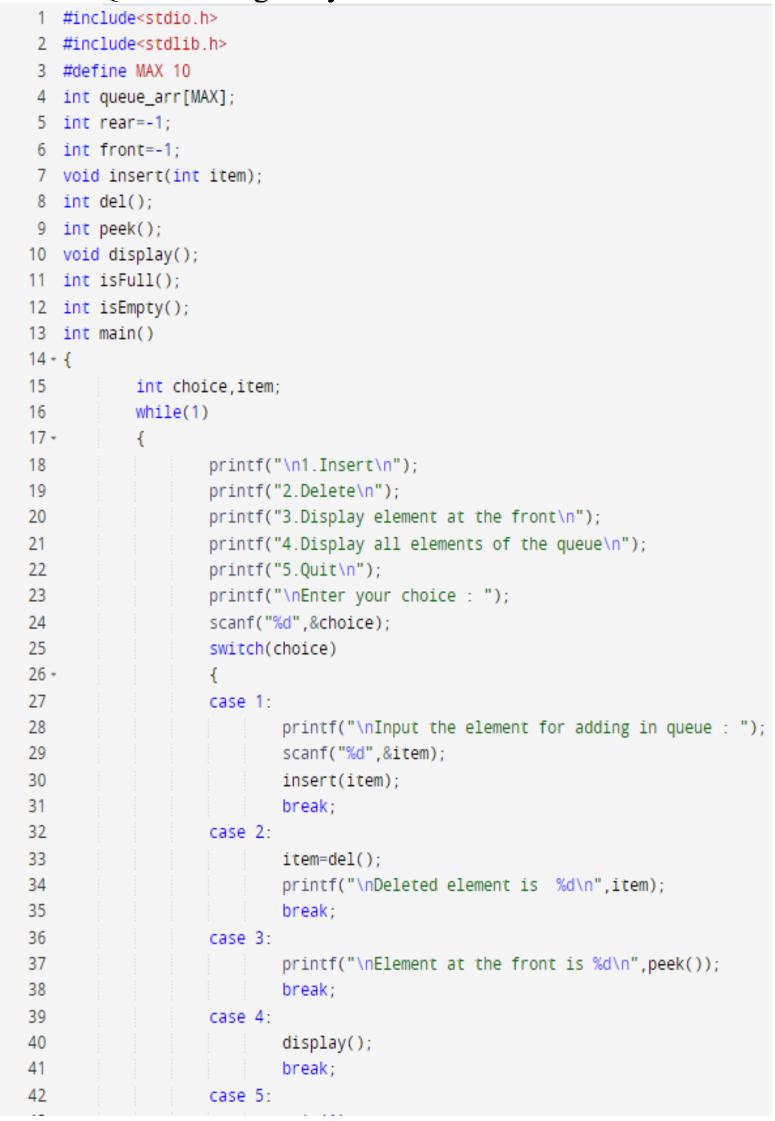
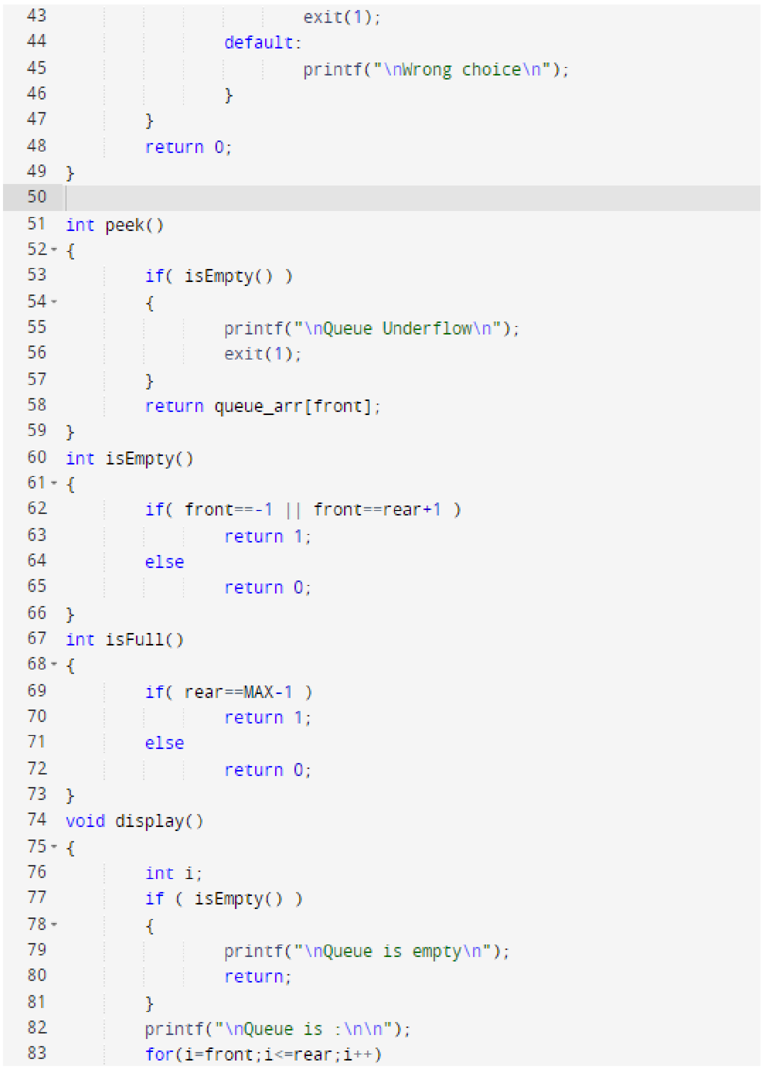
BHAVLEEN KAUR
102155010
3EIC-2

LAB ASSIGNMENT 6
Stacks and Queues

1. Write a menu driven program with 4 options (Insert, Delete, Display, and Exit) to demonstrate the working of Queues using arrays.

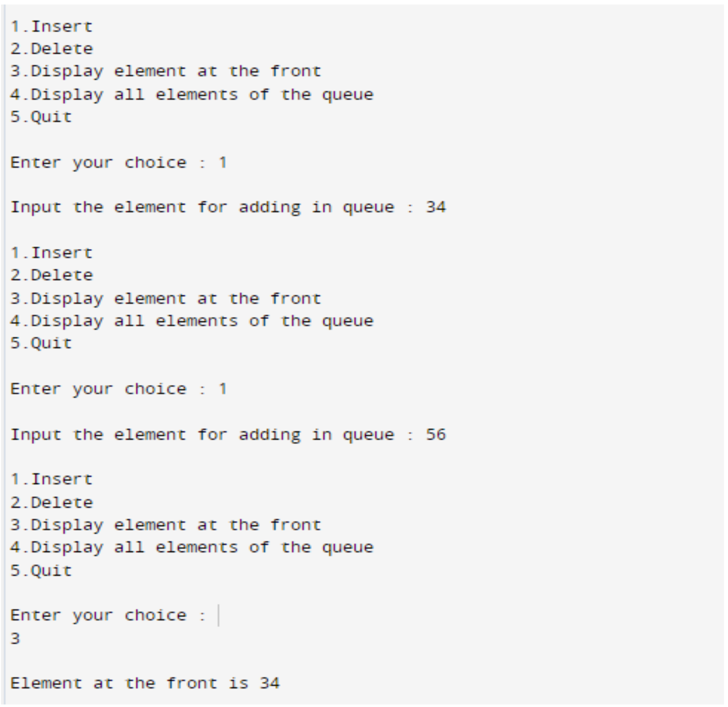


```c
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
int queue_arr[MAX];
int rear=-1;
int front=-1;
void insert(int item);
int del();
int peek();
void display();
int isFull();
int isEmpty();
int main()
{
        int choice,item;
        while(1)
        {
                printf("\n1.Insert\n");
                printf("2.Delete\n");
                printf("3.Display element at the front\n");
                printf("4.Display all elements of the queue\n");
                printf("5.Quit\n");
                printf("\nEnter your choice : ");
                scanf("%d",&choice);
                switch(choice)
                {
                case 1:
                        printf("\nInput the element for adding in queue : ");
                        scanf("%d",&item);
                        insert(item);
                        break;
                case 2:
                        item=del();
                        printf("\nDeleted element is  %d\n",item);
                        break;
                case 3:
                        printf("\nElement at the front is %d\n",peek());
                        break;
                case 4:
                        display();
                        break;
                case 5:
```



```c
                        exit(1);
                default:
                        printf("\nWrong choice\n");
                }
        }
        return 0;
}

int peek()
{
        if( isEmpty() )
        {
                printf("\nQueue Underflow\n");
                exit(1);
        }
        return queue_arr[front];
}
int isEmpty()
{
        if( front==-1 || front==rear+1 )
                return 1;
        else
                return 0;
}
int isFull()
{
        if( rear==MAX-1 )
                return 1;
        else
                return 0;
}
void display()
{
        int i;
        if ( isEmpty() )
        {
                printf("\nQueue is empty\n");
                return;
        }
        printf("\nQueue is :\n\n");
        for(i=front;i<=rear;i++)
```

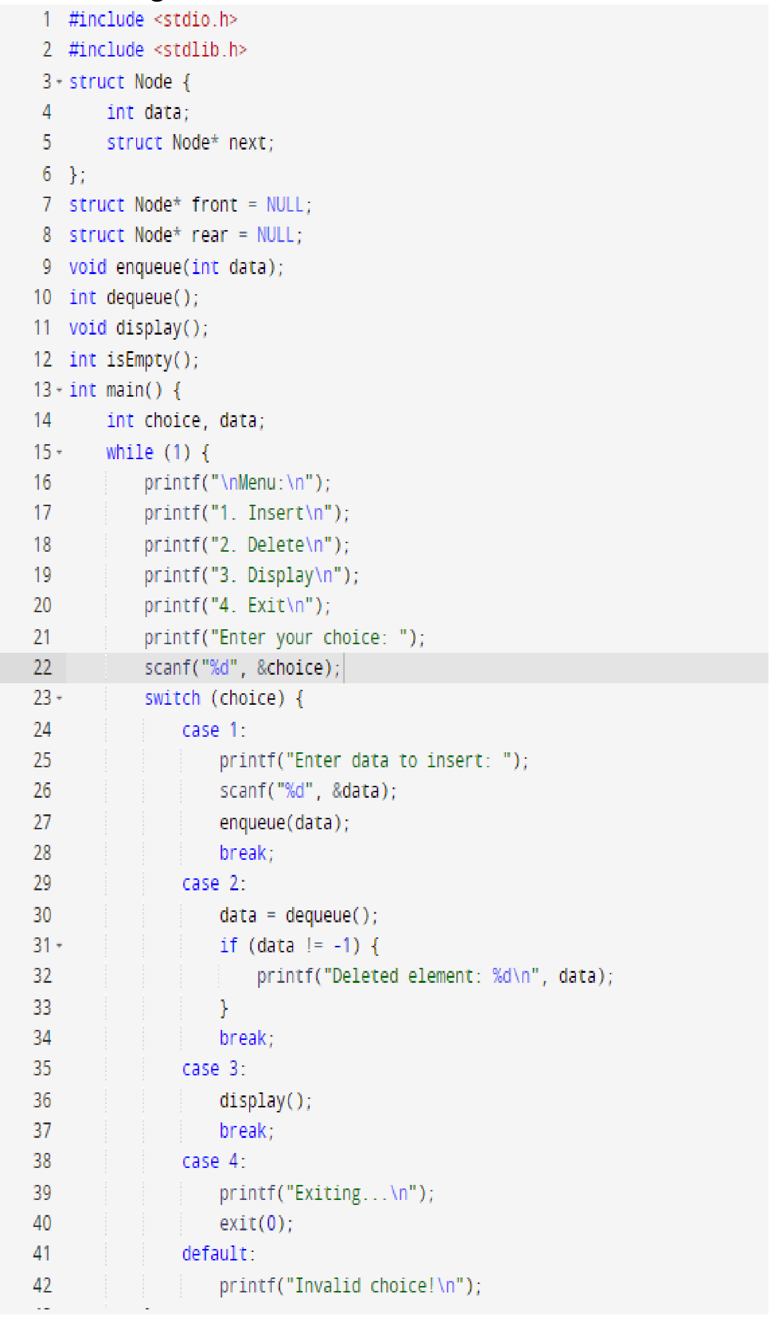
OUTPUT:



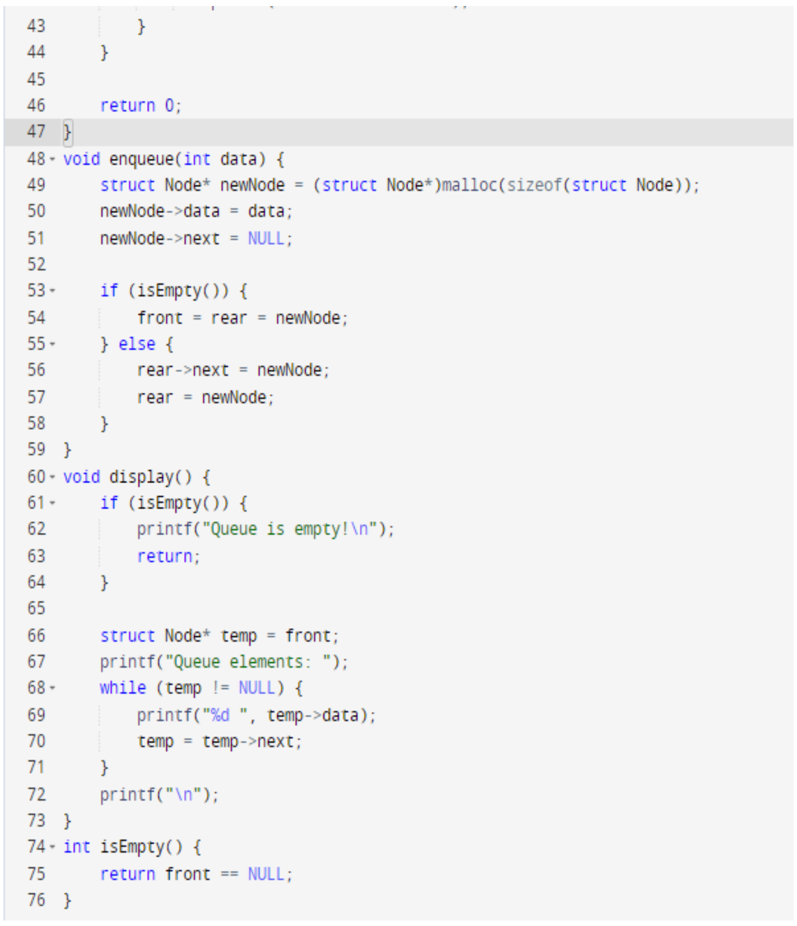
```
1.Insert
2.Delete
3.Display element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 1

Input the element for adding in queue : 34

1.Insert
2.Delete
3.Display element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 1

Input the element for adding in queue : 56

1.Insert
2.Delete
3.Display element at the front
4.Display all elements of the queue
5.Quit

Enter your choice : 
3

Element at the front is 34
```

BHAVLEEN KAUR
102155010
3EIC-2

2. Write a menu driven program with 4 options (Insert, Delete, Display, and Exit) to demonstrate the working of Queues using linked-list.



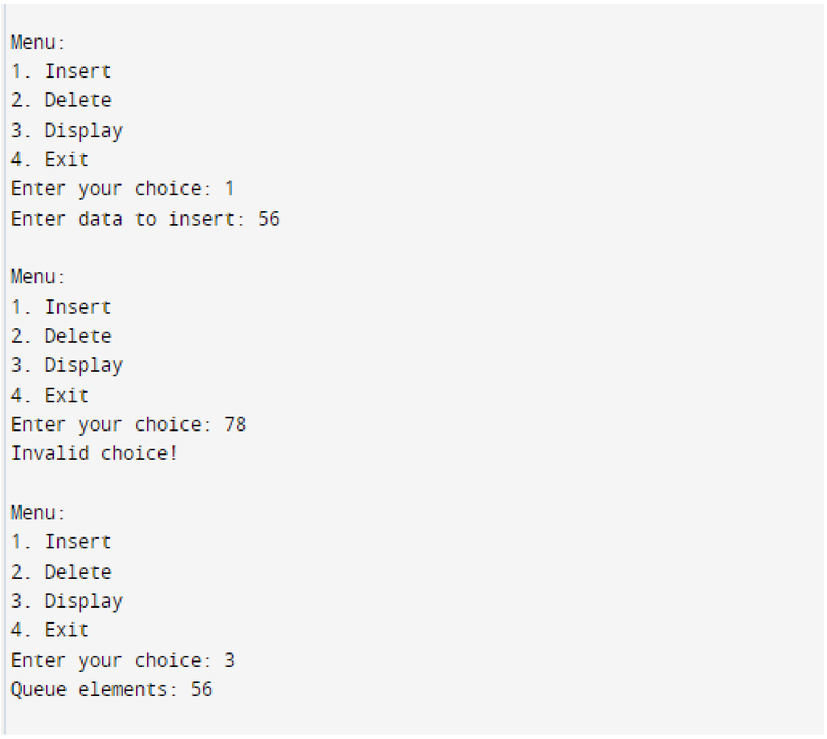
```c
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* next;
};
struct Node* front = NULL;
struct Node* rear = NULL;
void enqueue(int data);
int dequeue();
void display();
int isEmpty();
int main() {
    int choice, data;
    while (1) {
        printf("\nMenu:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                enqueue(data);
                break;
            case 2:
                data = dequeue();
                if (data != -1) {
                    printf("Deleted element: %d\n", data);
                }
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice!\n");
```



```c
        }
    }

    return 0;
}
void enqueue(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;

    if (isEmpty()) {
        front = rear = newNode;
    } else {
        rear->next = newNode;
        rear = newNode;
    }
}
void display() {
    if (isEmpty()) {
        printf("Queue is empty!\n");
        return;
    }

    struct Node* temp = front;
    printf("Queue elements: ");
    while (temp != NULL) {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int isEmpty() {
    return front == NULL;
}
```

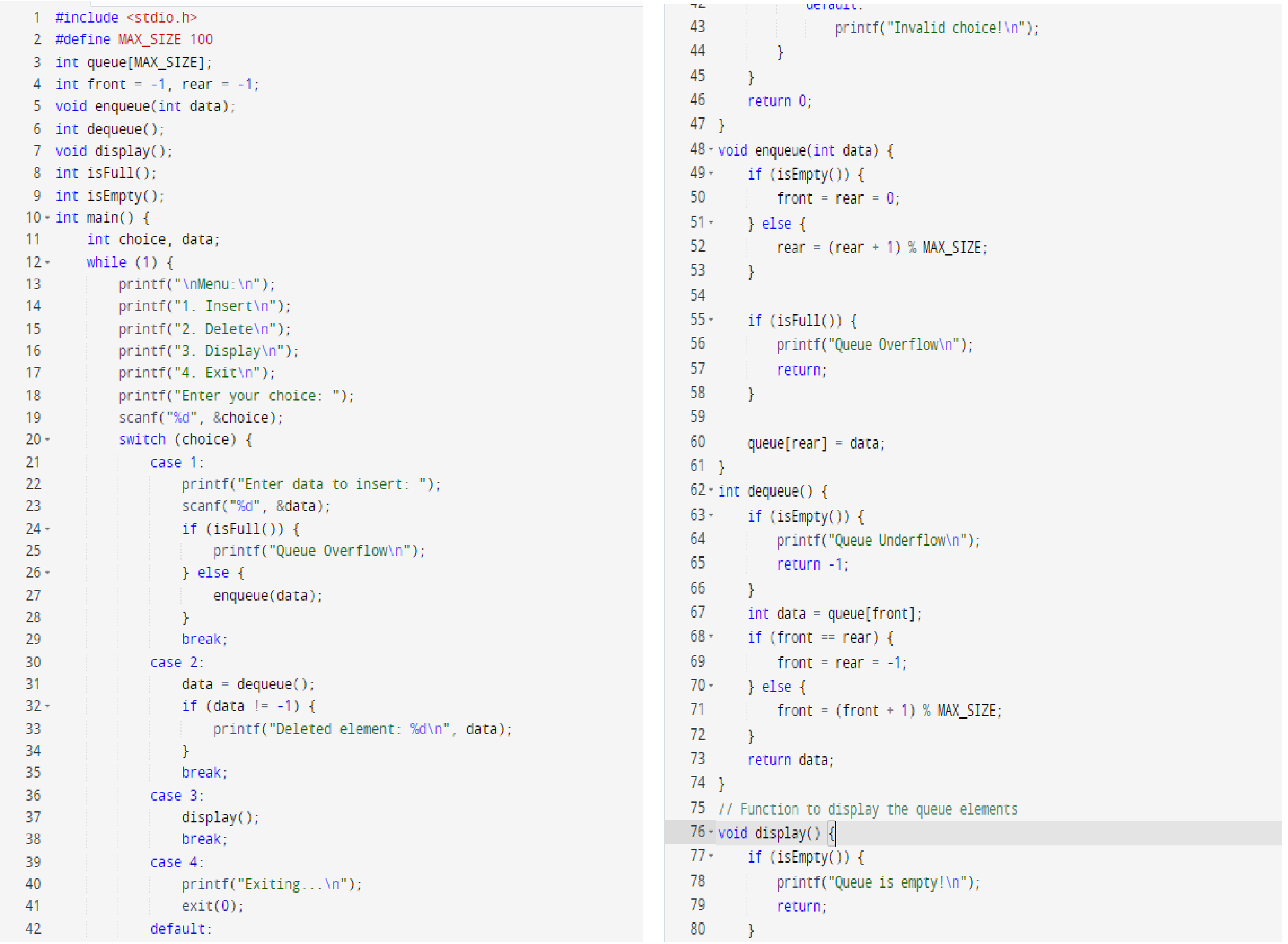
BHAVLEEN KAUR
102155010
3EIC-2

OUTPUT:



```
Menu:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 1
Enter data to insert: 56

Menu:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 78
Invalid choice!

Menu:
1. Insert
2. Delete
3. Display
4. Exit
Enter your choice: 3
Queue elements: 56
```

3. Write a menu driven program with 4 options (Insert, Delete, Display, and Exit) to demonstrate the working of Circular Queues (arrays)



```c
#include <stdio.h>
#define MAX_SIZE 100
int queue[MAX_SIZE];
int front = -1, rear = -1;
void enqueue(int data);
int dequeue();
void display();
int isFull();
int isEmpty();
int main() {
    int choice, data;
    while (1) {
        printf("\nMenu:\n");
        printf("1. Insert\n");
        printf("2. Delete\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice) {
            case 1:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                if (isFull()) {
                    printf("Queue Overflow\n");
                } else {
                    enqueue(data);
                }
                break;
            case 2:
                data = dequeue();
                if (data != -1) {
                    printf("Deleted element: %d\n", data);
                }
                break;
            case 3:
                display();
                break;
            case 4:
                printf("Exiting...\n");
                exit(0);
            default:
```

```c
                printf("Invalid choice!\n");
        }
    }
    return 0;
}
void enqueue(int data) {
    if (isEmpty()) {
        front = rear = 0;
    } else {
        rear = (rear + 1) % MAX_SIZE;
    }

    if (isFull()) {
        printf("Queue Overflow\n");
        return;
    }

    queue[rear] = data;
}
int dequeue() {
    if (isEmpty()) {
        printf("Queue Underflow\n");
        return -1;
    }
    int data = queue[front];
    if (front == rear) {
        front = rear = -1;
    } else {
        front = (front + 1) % MAX_SIZE;
    }
    return data;
}
// Function to display the queue elements
void display() {
    if (isEmpty()) {
        printf("Queue is empty!\n");
        return;
    }
```