ASSIGNMENT 13 >> PIPELINE

```
#include <iostream>
using namespace std;

#define MAX 5  // Maximum number of orders the pizza parlor can take

class PizzaParlor {
private:
    int orders[MAX];  // Array to store orders
    int front, rear;  // Front and rear indices for the circular queue

public:
    // Constructor to initialize the queue
    PizzaParlor() {
        front = -1;
        rear = -1;
    }

    // Function to check if the queue is full
    bool isFull() {
        return ((rear + 1) % MAX == front);
    }

    // Function to check if the queue is empty
    bool isEmpty() {
        return (front == -1);
    }

    // Function to place an order (enqueue)
    void placeOrder(int orderID) {
```

```
    if (isFull()) {
        cout << "Order queue is full! Cannot place more orders." << endl;
        return;
    }

    // If the queue is initially empty
    if (isEmpty()) {
        front = rear = 0;
    } else {
        rear = (rear + 1) % MAX;  // Circularly increment rear
    }

    orders[rear] = orderID;
    cout << "Order " << orderID << " has been placed." << endl;
}

// Function to serve an order (dequeue)
void serveOrder() {
    if (isEmpty()) {
        cout << "No orders to serve! Queue is empty." << endl;
        return;
    }

    cout << "Order " << orders[front] << " has been served." << endl;

    // If the queue has only one element
    if (front == rear) {
        front = rear = -1;  // Reset the queue to empty state
    } else {
        front = (front + 1) % MAX;  // Circularly increment front
    }
```

```
    }

    // Function to display all current orders
    void displayOrders() {
        if (isEmpty()) {
            cout << "No current orders. The queue is empty." << endl;
            return;
        }

        cout << "Current orders in the queue: ";
        int i = front;
        while (true) {
            cout << orders[i] << " ";
            if (i == rear) break;  // Stop when rear is reached
            i = (i + 1) % MAX;     // Circularly increment i
        }
        cout << endl;
    }
};

int main() {
    PizzaParlor pizzaParlor;
    int choice, orderID;

    do {
        cout << "\nPizza Parlor Menu:\n";
        cout << "1. Place Order\n";
        cout << "2. Serve Order\n";
        cout << "3. Display Current Orders\n";
        cout << "4. Exit\n";
        cout << "Enter your choice: ";
```

```
        cin >> choice;

        switch (choice) {
            case 1:
                cout << "Enter order ID: ";
                cin >> orderID;
                pizzaParlor.placeOrder(orderID);
                break;
            case 2:
                pizzaParlor.serveOrder();
                break;
            case 3:
                pizzaParlor.displayOrders();
                break;
            case 4:
                cout << "Exiting the system.\n";
                break;
            default:
                cout << "Invalid choice! Please try again.\n";
        }
    } while (choice != 4);

    return 0;
}
```

Output

Clear

```
/tmp/6h7LfIqdOZ.o

Pizza Parlor Menu:
1. Place Order
2. Serve Order
3. Display Current Orders
4. Exit
Enter your choice: 1
Enter order ID: 236
Order 236 has been placed.

Pizza Parlor Menu:
1. Place Order
2. Serve Order
3. Display Current Orders
4. Exit
Enter your choice: 2
Order 236 has been served.

Pizza Parlor Menu:
1. Place Order
2. Serve Order
3. Display Current Orders
4. Exit
Enter your choice: 3
No current orders. The queue is empty.
```