**Name : Prem Sah**
**PRN : 123B1B234**
**Div : D**

## Assignment no. 9

**b) Implement a checkout system for a supermarket to efficiently manage customer queues during peak hours. The system should support the following operations using a circular queue data structure:**

**a. Customer arrival**
**b. Customer checkout**
**c. Close Checkout Counter**
**d. View customer.**

```
#include <iostream>
#include <string>
class Customer {
public:
    std::string name;
Customer(const std::string& customerName) : name(customerName) {}
};
class CircularQueue {
private:
    Customer** queue;
    int front, rear, capacity, count;
public:
    CircularQueue(int size) : front(0), rear(-1), capacity(size), count(0) {
        queue = new Customer*[capacity];
    }
    ~CircularQueue() {
        for (int i = 0; i < count; i++) {
            delete queue[(front + i) % capacity];
        }
        delete[] queue;
```

```
}
void enqueue(const std::string& name) {
    if (count == capacity) {
        std::cout << "Queue full! " << name << " cannot join." << std::endl;
        return;
    }
    rear = (rear + 1) % capacity;
    queue[rear] = new Customer(name);
    count++;
    std::cout << name << " joined the queue." << std::endl;
}
void dequeue() {
    if (count == 0) {
        std::cout << "No customers to checkout." << std::endl;
        return;
    }
    std::cout << queue[front]->name << " checked out." << std::endl;
    delete queue[front];
    front = (front + 1) % capacity;
    count--;
}
void closeCounter() {
    while (count > 0) {
        dequeue();
    }
    std::cout << "Checkout counter closed." << std::endl;
}
void viewCustomers() const {
    if (count == 0) {
        std::cout << "No customers in the queue." << std::endl;
        return;
    }
    std::cout << "Customers in queue: ";
    for (int i = 0; i < count; i++) {
        std::cout << queue[(front + i) % capacity]->name << " ";
    }
    std::cout << std::endl;
```

```
    }
};

int main() {
    CircularQueue checkoutQueue(3); // Capacity of 3
    checkoutQueue.enqueue("Alice");
    checkoutQueue.enqueue("Bob");
    checkoutQueue.viewCustomers();
    checkoutQueue.dequeue();
    checkoutQueue.viewCustomers();
    checkoutQueue.enqueue("Charlie");
    checkoutQueue.enqueue("David"); // Should show full message
    checkoutQueue.viewCustomers();
    checkoutQueue.closeCounter(); // Close the checkout counter
      return 0;
}
```

**Output :**

```
Alice joined the queue.
Bob joined the queue.
Customers in queue: Alice Bob
Alice checked out.
Customers in queue: Bob
Charlie joined the queue.
David joined the queue.
Customers in queue: Bob Charlie David
Bob checked out.
Charlie checked out.
David checked out.
Checkout counter closed.
```

main.cpp

```cpp
#include <iostream>
#include <string>
class Customer {
public:
    std::string name;
Customer(const std::string& customerName) : name(customerName) {}
};
class CircularQueue {
private:
    Customer** queue;
    int front, rear, capacity, count;
public:
    CircularQueue(int size) : front(0), rear(-1), capacity(size), count(0) {
        queue = new Customer*[capacity];
    }
    ~CircularQueue() {
        for (int i = 0; i < count; i++) {
            delete queue[(front + i) % capacity];
        }
        delete[] queue;
    }
    void enqueue(const std::string& name) {
        if (count == capacity) {
            std::cout << "Queue full! " << name << " cannot join." << std::endl;
            return;
        }
        rear = (rear + 1) % capacity;
        queue[rear] = new Customer(name);
        count++;
        std::cout << name << " joined the queue." << std::endl;
    }
    void dequeue() {
        if (count == 0) {
            std::cout << "No customers to checkout." << std::endl;
            return;
        }
        std::cout << queue[front]->name << " checked out." << std::endl;
        delete queue[front];
        front = (front + 1) % capacity;
        count--;
    }
    void closeCounter() {
        while (count > 0) {
            dequeue();
        }
        std::cout << "Checkout counter closed." << std::endl;
    }
    void viewCustomers() const {
        if (count == 0) {
            std::cout << "No customers in the queue." << std::endl;
```

```
            return;
        }
        std::cout << "Customers in queue: ";
        for (int i = 0; i < count; i++) {
            std::cout << queue[(front + i) % capacity]->name << " ";
        }
        std::cout << std::endl;
    }
};

int main() {
    CircularQueue checkoutQueue(3); // Capacity of 3
    checkoutQueue.enqueue("Alice");
    checkoutQueue.enqueue("Bob");
    checkoutQueue.viewCustomers();
    checkoutQueue.dequeue();
    checkoutQueue.viewCustomers();
    checkoutQueue.enqueue("Charlie");
    checkoutQueue.enqueue("David"); // Should show full message
    checkoutQueue.viewCustomers();
    checkoutQueue.closeCounter(); // Close the checkout counter
     return 0;
}

```

Output

```
/tmp/5NDQjUuTWm.o
Alice joined the queue.
Bob joined the queue.
Customers in queue: Alice Bob
Alice checked out.
Customers in queue: Bob
Charlie joined the queue.
David joined the queue.
Customers in queue: Bob Charlie David
Bob checked out.
Charlie checked out.
David checked out.
Checkout counter closed.


=== Code Execution Successful ===
```