

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

```
1 #include <iostream>
2 #include <string>
3 class Customer {
4 public:
5     std::string name;
6     Customer(const std::string& customerName) : name(customerName) {}
7 };
8 class CircularQueue {
9 private:
10     Customer** queue;
11     int front, rear, capacity, count;
12 public:
13     CircularQueue(int size) : front(0), rear(-1), capacity(size), count(0) {
14         queue = new Customer*[capacity];
15     }
16     ~CircularQueue() {
17         for (int i = 0; i < count; i++) {
18             delete queue[(front + i) % capacity];
19         }
20         delete[] queue;
21     }
22     void enqueue(const std::string& name) {
23         if (count == capacity) {
24             std::cout << "Queue full! " << name << " cannot join." << std::endl;
25             return;
26         }
27         rear = (rear + 1) % capacity;
28         queue[rear] = new Customer(name);
29         count++;
30         std::cout << name << " joined the queue." << std::endl;
31     }
32     void dequeue() {
33         if (count == 0) {
34             std::cout << "No customers to checkout." << std::endl;
35             return;
36         }
37         std::cout << queue[front]->name << " checked out." << std::endl;
38         delete queue[front];
39         front = (front + 1) % capacity;
40         count--;
41     }
42     void closeCounter() {
43         while (count > 0) {
44             dequeue();
45         }
46         std::cout << "Checkout counter closed." << std::endl;
47     }
48     void viewCustomers() const {
49         if (count == 0) {
50             std::cout << "No customers in the queue." << std::endl;
51         }
52     }
```

```

51         return;
52     }
53     std::cout << "Customers in queue: ";
54     for (int i = 0; i < count; i++) {
55         std::cout << queue[(front + i) % capacity]->name << " ";
56     }
57     std::cout << std::endl;
58 }
59 };
60
61 int main() {
62     CircularQueue checkoutQueue(3); // Capacity of 3
63     checkoutQueue.enqueue("Alice");
64     checkoutQueue.enqueue("Bob");
65     checkoutQueue.viewCustomers();
66     checkoutQueue.dequeue();
67     checkoutQueue.viewCustomers();
68     checkoutQueue.enqueue("Charlie");
69     checkoutQueue.enqueue("David"); // Should show full message
70     checkoutQueue.viewCustomers();
71     checkoutQueue.closeCounter(); // Close the checkout counter
72     return 0;
73 }
74

```

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 ===