Data Structures Experiment 6

Name : Shubham Dorake
PRN : B24CE1042
Branch and Batch : Computer Engineering SY1 Batch (B)

Title : Coffee Shop Line (Simple Queue):
Problem Statement :
Arrival: Customers arrive at the coffee shop and stand in line. Order Processing: The first customer in line gets their order taken, and the barista starts making the coffee. Serving: Once the first customer is served, they leave the queue, and the next customer in line moves forward to be served. Write a program to implement a simple queue

---

CODE :

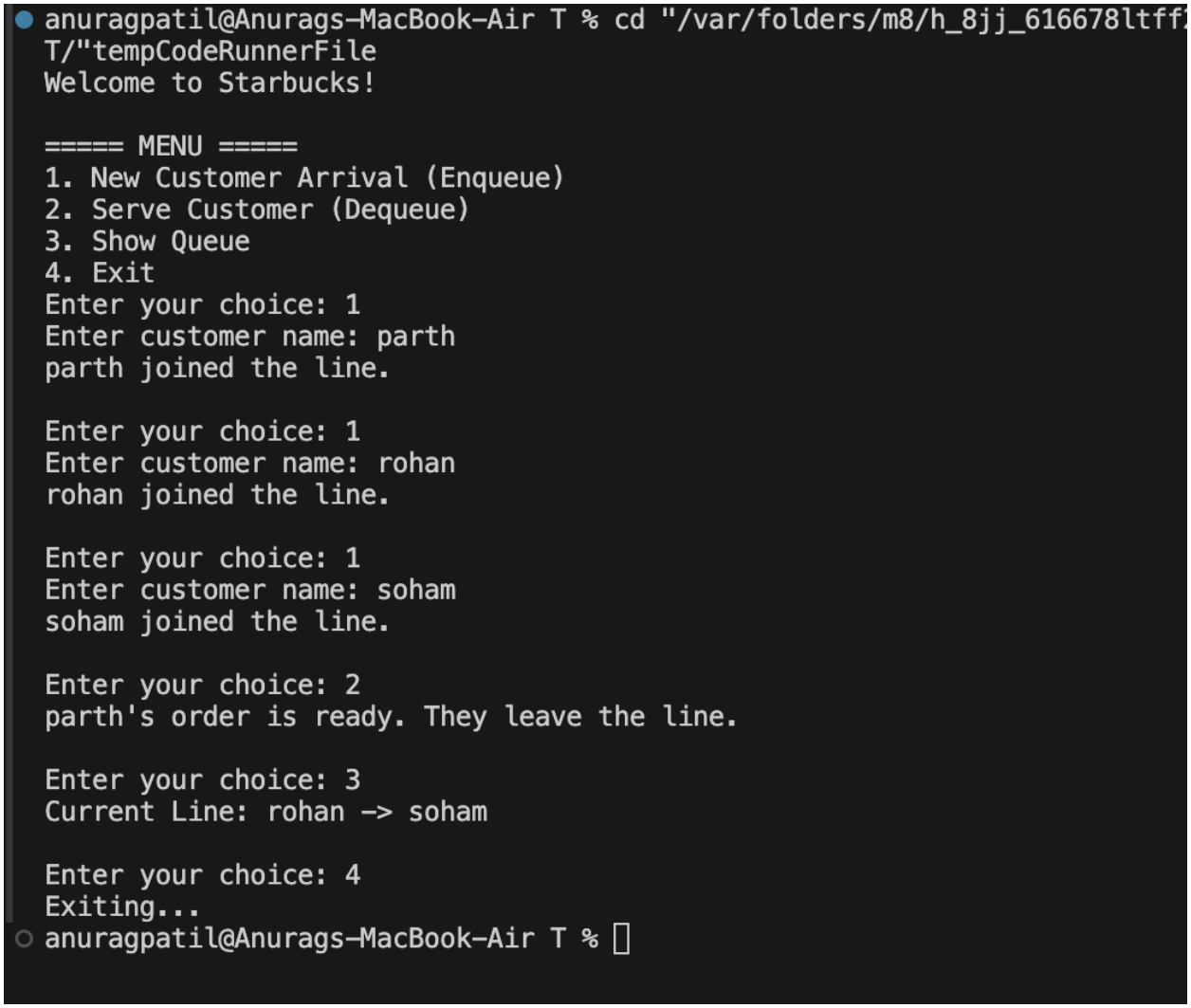
```
#include <iostream>
#include <string>
using namespace std;

struct Node {
    string name;
    Node* next;
};

class CustomerQueue {
private:
    Node* front;
    Node* rear;

public:
    CustomerQueue() {
        front = nullptr;
        rear = nullptr;
    }
```

```
~CustomerQueue() {
    while (front != nullptr) {
        Node* temp = front;
        front = front->next;
        delete temp;
    }
}

void enqueue(string customerName) {
    Node* temp = new Node();
    temp->name = customerName;
    temp->next = nullptr;

    if (rear == nullptr) {
        front = rear = temp;
    } else {
        rear->next = temp;
        rear = temp;
    }
    cout << customerName << " joined the line.\n";
}

void dequeue() {
    if (front == nullptr) {
        cout << "No customers in line.\n";
        return;
    }
    Node* temp = front;
    cout << front->name << "'s order is ready. They leave the line.\n";
    front = front->next;
    if (front == nullptr) {
        rear = nullptr;
    }
    delete temp;
}

void display() {
    if (front == nullptr) {
        cout << "Line is empty.\n";
```

```
            return;
        }
        cout << "Current Line: ";
        Node* temp = front;
        while (temp != nullptr) {
            cout << temp->name;
            if (temp->next != nullptr)
                cout << " -> ";
            temp = temp->next;
        }
        cout << "\n";
    }
};

int main() {
    int choice;
    CustomerQueue q;
    string name;

    cout << "Welcome to Starbucks!\n";

    cout << "\n===== MENU =====";
    cout << "\n1. New Customer Arrival (Enqueue)";
    cout << "\n2. Serve Customer (Dequeue)";
    cout << "\n3. Show Queue";
    cout << "\n4. Exit";

    do {
        cout << "\nEnter your choice: ";
        cin >> choice;
        cin.ignore(numeric_limits<streamsize>::max(), '\n'); // clear buffer

        switch (choice) {
            case 1:
                cout << "Enter customer name: ";
                getline(cin, name); // allows full names
                q.enqueue(name);
                break;

            case 2:
```

```
            q.dequeue();
            break;

        case 3:
            q.display();
            break;

        case 4:
            cout << "Exiting...\n";
            break;

        default:
            cout << "Invalid choice. Please try again.\n";
            break;
      }
   } while (choice != 4);

   return 0;
}
```

OUTPUT :


anuragpatil@Anurags-MacBook-Air T % cd "/var/folders/m8/h_8jj_616678ltff
T/"tempCodeRunnerFile
Welcome to Starbucks!
===== MENU =====
1. New Customer Arrival (Enqueue)
2. Serve Customer (Dequeue)
3. Show Queue
4. Exit
Enter your choice: 1
Enter customer name: parth
parth joined the line.
Enter your choice: 1
Enter customer name: rohan
rohan joined the line.
Enter your choice: 1
Enter customer name: soham
soham joined the line.
Enter your choice: 2
parth's order is ready. They leave the line.
Enter your choice: 3
Current Line: rohan -> soham
Enter your choice: 4
Exiting...
anuragpatil@Anurags-MacBook-Air T %

Title : Printer Spooler (Circular Queue):

Problem Statement :

In a multi-user environment, printers often use a circular queue to manage print jobs. Each print job is added to the queue, and the printer processes them in the order they arrive. Once a print job is completed, it moves out of the queue, and the next job is processed, efficiently managing the flow of print tasks. Implement the Printer Spooler system using a circular queue without using built-in queues.

CODE :

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

class CircularQueue {
    int front, rear, capacity;
    string* queue;

public:
    CircularQueue(int size) {
        capacity = size;
        queue = new string[capacity];
        front = -1;
        rear = -1;
    }

    bool isFull() {
        return (front == 0 && rear == capacity - 1) || ((rear + 1) % capacity == front);
    }

    bool isEmpty() {
        return front == -1;
    }

    void enqueue(string jobName) {
```

```
    if (isFull()) {
        cout << "Printer queue is full! Cannot add job: " << jobName << endl;
        return;
    }
    if (isEmpty()) {
        front = 0;
        rear = 0;
    } else {
        rear = (rear + 1) % capacity;
    }
    queue[rear] = jobName;
    cout << "Print job '" << jobName << "' added to the queue.\n";
}

void dequeue() {
    if (isEmpty()) {
        cout << "Printer queue is empty! No job to process.\n";
        return;
    }
    cout << "Processing and removing print job: " << queue[front] << endl;
    if (front == rear) {
        front = rear = -1;
    } else {
        front = (front + 1) % capacity;
    }
}

void display() {
    if (isEmpty()) {
        cout << "Printer queue is empty.\n";
        return;
    }
    cout << "Current print queue: ";
    int i = front;
    while (true) {
        cout << queue[i] << " ";
        if (i == rear)
            break;
        i = (i + 1) % capacity;
    }
```

```
        cout << endl;
    }
};

int main() {
    int size;
    cout << "------ Printer Spooler ------ ";
    cout << "\nEnter printer queue capacity: ";
    cin >> size;
    CircularQueue printer(size);

    int choice;
    string jobName;
    do {
        cout << "\n1. Add Print Job\n2. Process Print Job\n3. Display Queue\n4. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;
        switch (choice) {
            case 1:
                cout << "Enter print job name: ";
                cin >> jobName;
                printer.enqueue(jobName);
                break;
            case 2:
                printer.dequeue();
                break;
            case 3:
                printer.display();
                break;
            case 4:
                cout << "Exiting printer spooler.\n";
                break;
            default:
                cout << "Invalid choice! Try again.\n";
        }
    } while (choice != 4);

    return 0;
}
```

OUTPUT :

```
anuragpatil@Anurags-MacBook-Air T % cd "/var/folders/m8/h_8jj_616678ltf
T/"tempCodeRunnerFile
------ Printer Spooler ------
Enter printer queue capacity: 3

1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 1
Enter print job name: report
Print job 'report' added to the queue.

1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 1
Enter print job name: document
Print job 'document' added to the queue.

1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 2
Processing and removing print job: report

1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 3
Current print queue: document

1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 4
Exiting printer spooler.
anuragpatil@Anurags-MacBook-Air T %
```

Thank You !!!