**Name - Srushti Bhivaji Salgar**
**PRN - B24CE1079**
**Class - SE 2 Batch - A**
**Subject - Data Structures**
**Assignment 6**

**Coffee Shop Line (Simple queue)**

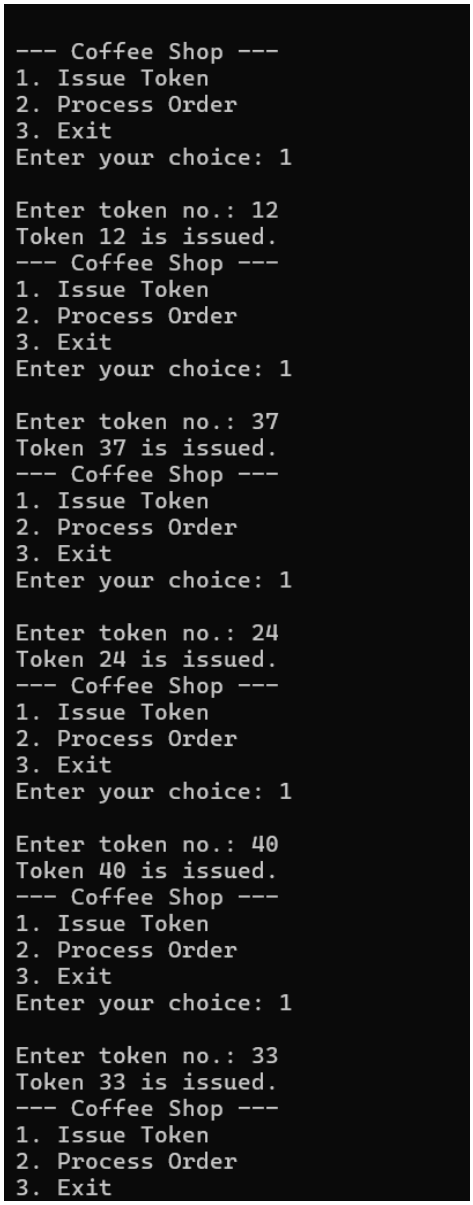
/*PROBLEM STATEMENT:
Coffee Shop Line (Simple Queue):
Arrival: Customers arrive at the coffee shop and stand in line. Order Processing: The first customer in line gets their order taken, and the barista starts making the coffee.Serving: Once the first customer is served, they leave the queue, and the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

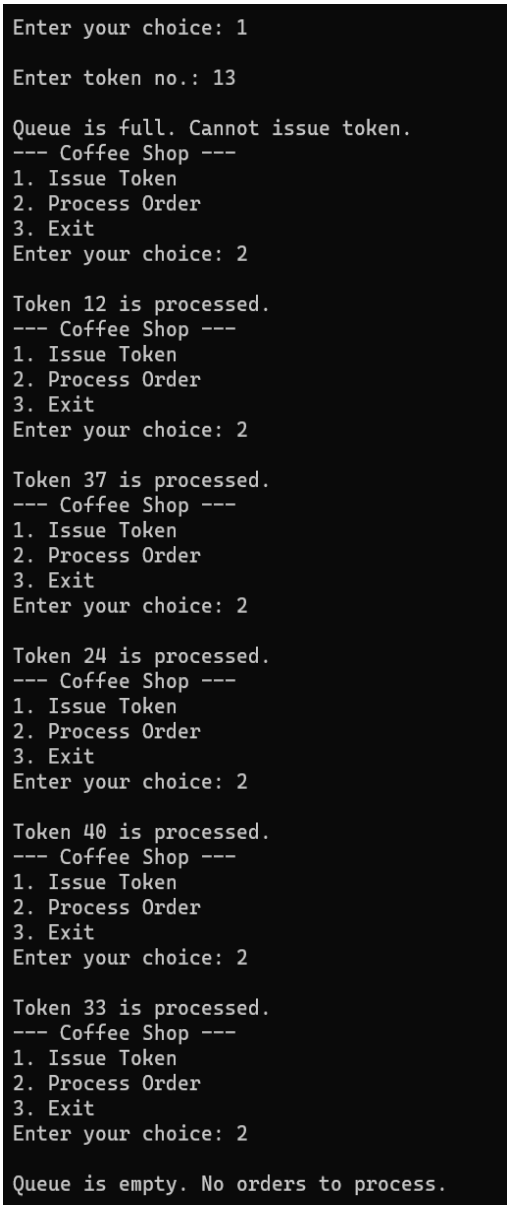
```
#include <iostream>
using namespace std;

class shop {
    int size = 5;
    int token_shop[5];
    int r = -1; // rear
    int f = 0; // front

public:
    void Enqueue(int t);
    int Dequeue();
    int isEmpty();
    int isFull();
};

int shop::isEmpty() {
    return (f > r);
}

int shop::isFull() {
    return (r == size - 1);
}
void shop::Enqueue(int t) {
    if (isFull()) {
        cout << "\nQueue is full. Cannot issue token.";
    } else {
        r++;
```

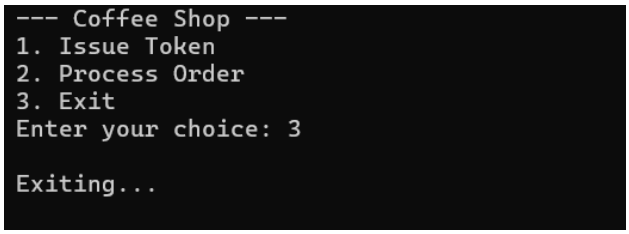
```
        token_shop[r] = t;
        cout << "Token " << t << " is issued.";
    }
}

int shop::Dequeue() {
    if (isEmpty()) {
        cout << "\nQueue is empty. No orders to process.";
        return -1;
    } else {
        int x = token_shop[f];
        f++;
        cout << "\nToken " << x << " is processed.";
        return x;
    }
}
int main() {
    shop s;
    int t; // token no.
    int choice;

    do {
        cout << "\n--- Coffee Shop ---";
        cout << "\n1. Issue Token";
        cout << "\n2. Process Order";
        cout << "\n3. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "\nEnter token no.: ";
            cin >> t;
            s.Enqueue(t);
            break;
        case 2:
            s.Dequeue();
            break;
        case 3:
            cout << "\nExiting...";
            break;

        default:
            cout << "\nInvalid choice. Please try again.";}
```

```
    } while (choice != 3);

    return 0;
}
```

OUTPUT



**Printer Spooler (Circular queue)**

/*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.*/

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

class printer {
    int size = 5;
    int jobs[5];
    int r = -1; // rear
    int f = -1; // front

public:
    void Enqueue(int t);
    int Dequeue();
    int isEmpty();
    int isFull();
    void Display();
};

int printer::isEmpty() {
    return (f == -1);
}

int printer::isFull() {
    return ((r + 1) % size == f);
}

void printer::Enqueue(int t) {
    if (isFull()) {
        cout << "\nPrinter queue is full. Cannot add new job.";
    } else {
        if (isEmpty()) {
            f = 0;
            r = 0;
        } else {
            r = (r + 1) % size;
        }
        jobs[r] = t;
        cout << "Print Job " << t << " is added to the queue.";
```

```
    }
}

int printer::Dequeue() {
    if (isEmpty()) {
        cout << "\nPrinter queue is empty. No jobs to process.";
        return -1;
    } else {
        int x = jobs[f];
        if (f == r) {
            // only one job left
            f = -1;
            r = -1;
        } else {
            f = (f + 1) % size;
        }
        cout << "\nPrint Job " << x << " is processed.";
        return x;
    }
}

void printer::Display() {
    if (isEmpty()) {
        cout << "\nNo pending print jobs.";
    } else {
        cout << "\nCurrent Print Jobs in Queue: ";
        int i = f;
        while (true) {
            cout << jobs[i] << " ";
            if (i == r)
                break;
            i = (i + 1) % size;
        }
        cout << endl;
    }
}

int main() {
    printer p;
    int t; // job id
    int choice;

    do {
        cout << "\n--- Printer Spooler (Circular Queue) ---";
```

```
        cout << "\n1. Add Print Job";
        cout << "\n2. Process Print Job";
        cout << "\n3. Display Queue";
        cout << "\n4. Exit";
        cout << "\nEnter your choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "\nEnter Print Job ID: ";
            cin >> t;
            p.Enqueue(t);
            break;

        case 2:
            p.Dequeue();
            break;

        case 3:
            p.Display();
            break;

        case 4:
            cout << "\nExiting Printer Spooler...";
            break;

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

    return 0;
}
```

## OUTPUT

```
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 1

Enter Print Job ID: 790
Print Job 790 is added to the queue.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 1

Enter Print Job ID: 273
Print Job 273 is added to the queue.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 1

Enter Print Job ID: 228
Print Job 228 is added to the queue.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 1

Enter Print Job ID: 188

Printer queue is full. Cannot add new job.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
```

```
Enter your choice: 2

Print Job 790 is processed.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice:
2

Print Job 273 is processed.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 2

Print Job 228 is processed.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 2

Printer queue is empty. No jobs to process.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 3

No pending print jobs.
--- Printer Spooler (Circular Queue) ---
1. Add Print Job
2. Process Print Job
3. Display Queue
4. Exit
Enter your choice: 4

Exiting Printer Spooler...
```