**Queue Simulation – Ticket Counter System**

**Abstract:**

This project simulates a real-world queue system such as a ticket counter or bank queue using the Queue Data Structure. It follows the First In, First Out (FIFO) concept, where customers are served in the order they arrive. The program allows users to add customers to the queue, serve them, and display the current queue. This project helps in understanding the practical application of queues in real-time systems.

**Introduction:**

A **Queue** is a linear data structure that stores elements in a sequence and allows operations in a specific order — **First In, First Out (FIFO)**.  
In real life, queues are used in systems like:

* Ticket booking counters
* Bank service lines
* CPU task scheduling
* Call center systems

This project models such a system where customers enter a queue, wait for their turn, and get served in the same order.

**Existing System:**

In the traditional system:

* Queues are managed manually.
* It is difficult to track the exact order or count of customers.
* Managing large crowds or service requests is inefficient.
* Errors or confusion may occur due to lack of systemization.

Hence, the existing manual process is **time-consuming and unorganized**.

**Proposed System:**

The proposed system uses a **Queue Data Structure** to handle customer flow efficiently and automatically.

**Features:**

* Add (Enqueue) a customer to the queue
* Serve (Dequeue) the first customer
* Display current queue status
* Exit the system

This simulation is a simple but effective demonstration of queue operations and can be extended to more complex queue-based applications like customer service software or online ticket booking systems.

**Software Requirements:**

* **Programming Language:** Java
* **IDE:** Eclipse / IntelliJ IDEA / VS Code
* **JDK Version:** 8 or above
* **Operating System:** Windows / Linux / macOS

**Hardware Requirements:**

* **Processor:** Intel Core i3 or higher
* **RAM:** Minimum 2 GB
* **Hard Disk:** 500 MB free space
* **Input Device:** Keyboard
* **Output Device:** Monitor

**Source Code**

#include <stdio.h>

#include <stdlib.h>

#define SIZE 10 // Maximum number of customers in the queue

int queue[SIZE];

int front = -1, rear = -1;

// Function to add customer to the queue

void enqueue(int customerID) {

if (rear == SIZE - 1) {

printf("Queue is full! Customer %d cannot join the queue.\n", customerID);

} else {

if (front == -1)

front = 0;

rear++;

queue[rear] = customerID;

printf("Customer %d joined the queue.\n", customerID);

}

}

// Function to remove customer (serve)

void dequeue() {

if (front == -1 || front > rear) {

printf("No customers in the queue.\n");

} else {

printf("Customer %d got the ticket and left the counter.\n", queue[front]);

front++;

if (front > rear)

front = rear = -1;

}

}

// Function to display queue

void displayQueue() {

if (front == -1)

printf("Queue is empty.\n");

else {

printf("Current Queue: ");

for (int i = front; i <= rear; i++)

printf("%d ", queue[i]);

printf("\n");

}

}

int main() {

int choice, customerID = 1;

while (1) {

printf("\n--- Ticket Counter System ---\n");

printf("1. New Customer Arrives (Enqueue)\n");

printf("2. Serve Customer (Dequeue)\n");

printf("3. Display Queue\n");

printf("4. Exit\n");

printf("Enter your choice: ");

scanf("%d", &choice);

switch (choice) {

case 1:

enqueue(customerID++);

break;

case 2:

dequeue();

break;

case 3:

displayQueue();

break;

case 4:

printf("Exiting the system. Thank you!\n");

exit(0);

default:

printf("Invalid choice! Try again.\n");

}

}

return 0;

}

**Sample output** **:**

--- Ticket Counter System ---

1. New Customer Arrives (Enqueue)

2. Serve Customer (Dequeue)

3. Display Queue

4. Exit

Enter your choice: 1

Customer 1 joined the queue.

Enter your choice: 1

Customer 2 joined the queue.

Enter your choice: 3

Current Queue: 1 2

Enter your choice: 2

Customer 1 got the ticket and left the counter.

Enter your choice: 3

Current Queue: 2

**Conclusion:**

The **Queue Simulation System** successfully demonstrates how the **Queue Data Structure** operates in a real-world scenario. It shows how customers can be efficiently managed using the **FIFO principle**.  
This project helps in understanding the core functionality of queues and serves as a foundation for more advanced applications such as **CPU scheduling**, **network packet handling**, and **real-time customer service management**.