**Lab 4**

**Objectives:**

The objective of this lab is to understand and implement M/M/1 queueing models using C programming to analyze different queuing scenarios. The lab focuses on simulating real-world applications of queuing theory, including a bank service system and a football stadium ticket counter. By implementing these simulations, students will compute key performance metrics such as the probability of no waiting time, expected number of customers in the system, and total time spent in the queue. Additionally, the lab includes writing a program to calculate important measures of an M/M/1 queue, such as system utilization, average queue length, and waiting time, for given values of arrival rate (λ) and service rate (μ). Through this lab, students will gain practical insights into the efficiency of single-server queues and how queuing models apply to real-life service systems.

1. **Customers arrive in a bank according to a Poisson's process with mean inter arrival time of 10 minutes. Customers spend an average of 5 minutes on the single available counter, and leave. Write a program in C to find:**
2. **Probability that a customer will not have to wait at the counter.**
3. **Expected number of customers in the bank.**
4. **Time a customer expects to spend in the bank.**

**Source Code:**

#include <stdio.h>

int main() {

double arrival\_time, service\_time, lambda, mu, rho, P0, L, W;

printf("Enter the mean inter-arrival time (in minutes): ");

scanf("%lf", &arrival\_time);

printf("Enter the average service time per customer (in minutes): ");

scanf("%lf", &service\_time);

lambda = 1.0 / arrival\_time; // Arrival rate (customers per minute)

mu = 1.0 / service\_time; // Service rate (customers per minute)

// Check if system is stable (rho < 1)

if (lambda >= mu) {

printf("\nSystem is unstable! Arrival rate must be less than service rate.\n");

return 1;

}

// Compute metrics

rho = lambda / mu; // Utilization factor

P0 = 1.0 - rho; // Probability of no waiting

L = lambda / (mu - lambda); // Expected number of customers in the bank

W = 1.0 / (mu - lambda); // Expected time a customer spends in the bank

// Print results

printf("\n=== Bank Queue System Analysis ===\n");

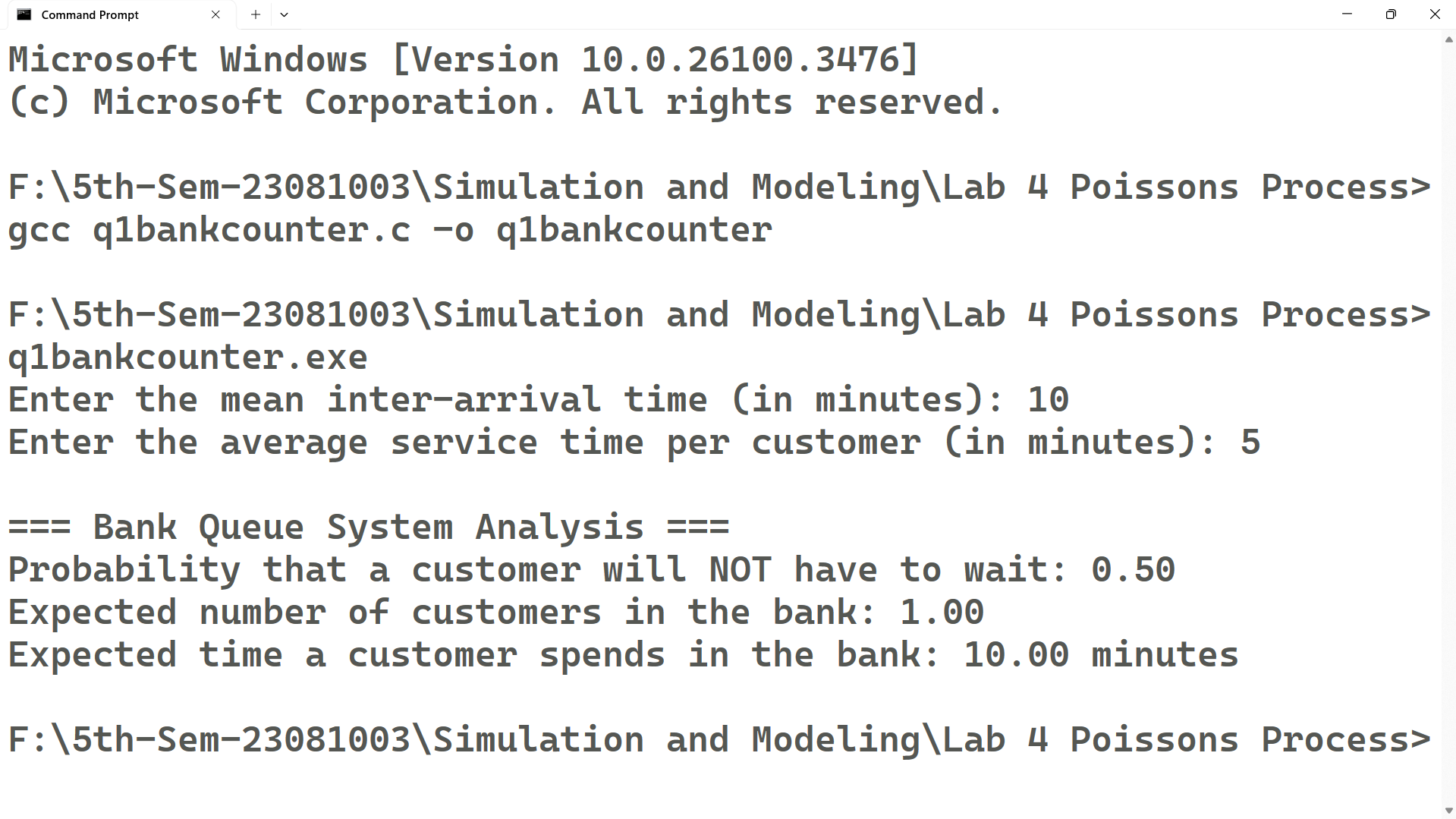
printf("Probability that a customer will NOT have to wait: %.2lf\n", P0);

printf("Expected number of customers in the bank: %.2lf\n", L);

printf("Expected time a customer spends in the bank: %.2lf minutes\n", W);

return 0;

}

**Output:**

1. **At the ticket counter of football stadium, people come in queue and purchase tickets. Arrival rate of customers is 1/min. It takes at the average 20 seconds to purchase the ticket. WAP in C to calculate total time spent by a sports fan to be seated in his seat, if it takes 1.5 minutes to reach the correct seat after purchasing the ticket. If a fan comes exactly before 2 minutes before the game starts, can sports fan expect to be seated for the kick-off?**

**Source Code:**

#include <stdio.h>

int main() {

// Given data

double lambda = 1.0; // Arrival rate (customers per minute)

double mu = 3.0; // Service rate (customers per minute)

double seat\_time = 1.5; // Time to reach the seat after purchase

// Utilization factor (traffic intensity)

double rho = lambda / mu;

// Expected number of customers in the system (L)

double L = lambda / (mu - lambda);

// Expected time in the system (waiting + service time)

double W = 1.0 / (mu - lambda);

// Total time spent before being seated

double total\_time = W + seat\_time;

// Print results

printf("=== Football Stadium Ticket Queue Analysis ===\n");

printf("Probability of immediate service: %.2lf\n", 1 - rho);

printf("Expected number of people in the system: %.2lf\n", L);

printf("Time spent in queue + service time: %.2lf minutes\n", W);

printf("Total time to be seated: %.2lf minutes\n", total\_time);

// Check if the fan will be seated before kickoff

if (total\_time <= 2.0)

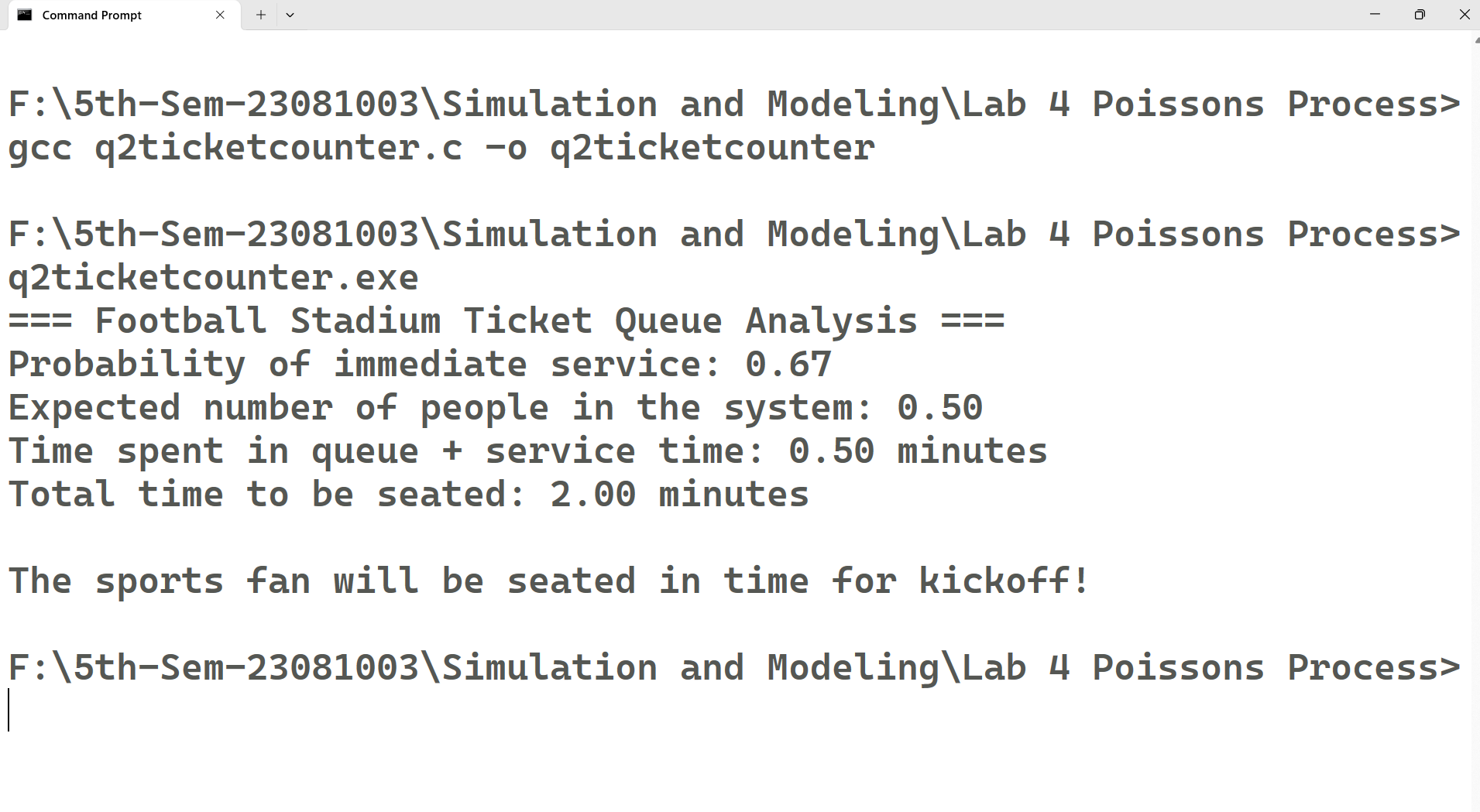
printf("\nThe sports fan will be seated in time for kickoff!\n");

else

printf("\nThe sports fan may miss the kickoff.\n");

return 0;

}

**Output:**

1. **Write a program to calculate measures of a M/M/1 Queue for a given value of Arrival Rate and Service Rate.**

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

void mm1\_queue(double lambda, double mu, int n) {

// Check system stability (lambda must be less than mu)

if (lambda >= mu) {

cout << "Error: Arrival rate (lambda) must be less than service rate (mu) for a stable system.\n";

return;

}

double rho = lambda / mu; // Server utilization (Traffic Intensity)

double Lq = pow(rho, 2) / (1 - rho); // Average number of customers in the queue

double Wq = Lq / lambda; // Average time a customer spends in the queue

double L = lambda / (mu - lambda); // Average number of customers in the system

double W = 1 / (mu - lambda); // Average time a customer spends in the system

double P0 = 1 - rho; // Probability of zero customers in the system

double Pn = P0 \* pow(rho, n); // Probability of exactly 'n' customers in the system

// Display results

cout << "\n=== M/M/1 Queue Metrics ===\n";

cout << "Traffic Intensity (?) = " << rho << endl;

cout << "Expected number of customers in queue (Lq) = " << Lq << endl;

cout << "Expected time in queue (Wq) = " << Wq << " minutes\n";

cout << "Expected number of customers in system (L) = " << L << endl;

cout << "Expected time in system (W) = " << W << " minutes\n";

cout << "Probability of zero customers in system (P0) = " << P0 << endl;

cout << "Probability of exactly " << n << " customers in system (Pn) = " << Pn << endl;

cout << "Server Utilization = " << rho << " (should be < 1 for stability)\n";

}

int main() {

double lambda, mu;

int n;

// Get user inputs

cout << "Enter the Arrival Rate (customers per minute): ";

cin >> lambda;

cout << "Enter the Service Rate (customers per minute): ";

cin >> mu;

cout << "Enter the number of jobs in the system (n): ";

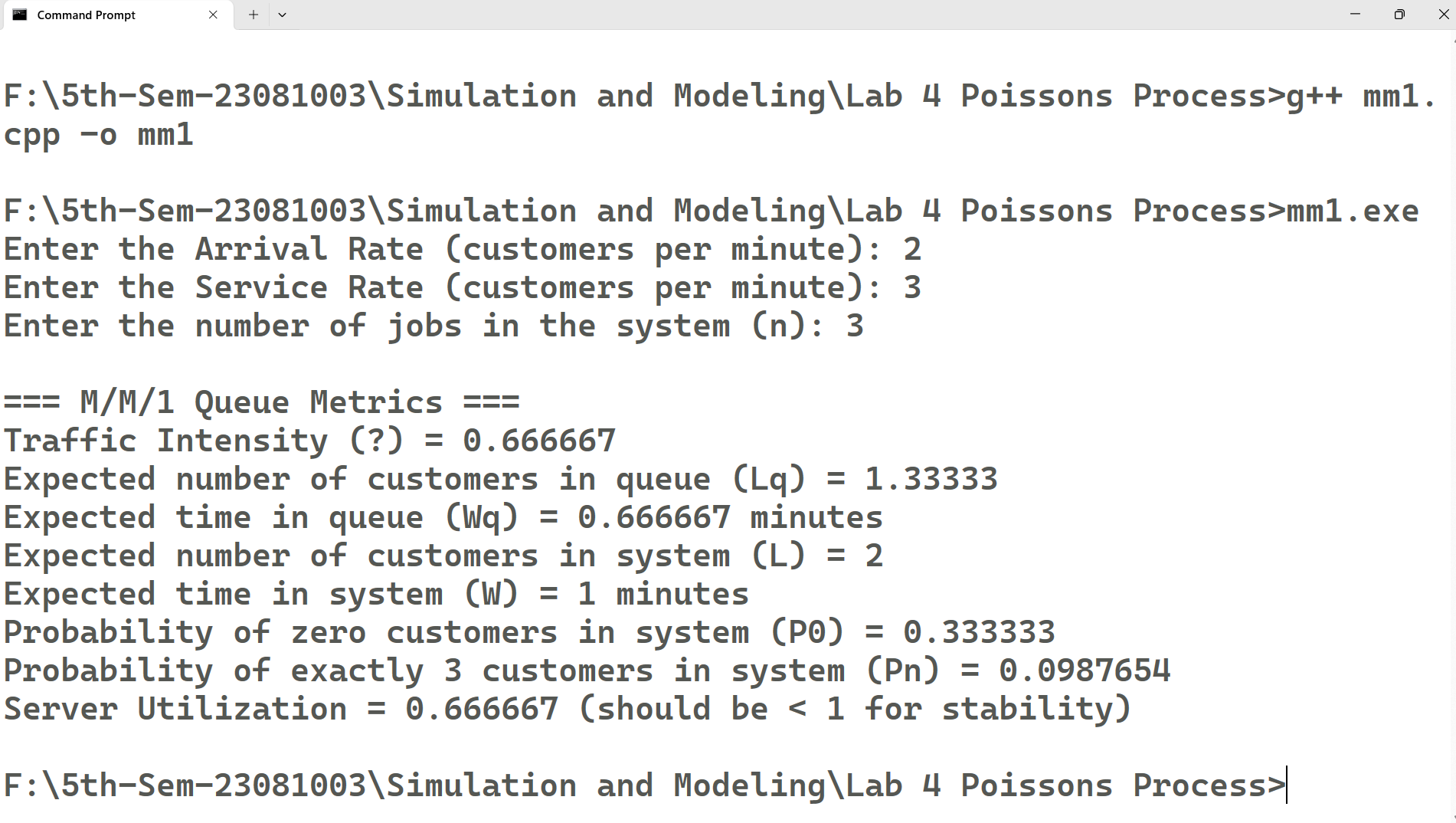
cin >> n;

// Call function to calculate M/M/1 queue metrics

mm1\_queue(lambda, mu, n);

return 0;

}

**Output:**

**Conclusion:**

In conclusion, this lab provided a practical understanding of M/M/1 queueing models and their real-world applications through C programming. By simulating a bank service system and a football stadium ticket counter, we successfully analyzed key performance metrics such as the probability of a customer not waiting, the expected number of customers in the system, and the total time spent in the queue. Additionally, implementing a general M/M/1 queue program allowed us to compute essential parameters like system utilization, queue length, and waiting time for given arrival and service rates. These experiments reinforced the significance of queueing theory in optimizing service efficiency and minimizing wait times in various domains, such as banking, ticketing, and customer service. Overall, this lab enhanced our understanding of queueing models and their practical implementation in computational simulations.