Simulation of a Single-Server Queueing System

Problem statement

This experiment involves the simulation of a single-server queue with a FIFO (First In, First Out) queuing discipline, also known as an M/M/1 queue. In this system, both the customer inter-arrival times and the service times are exponentially distributed, and there is only one server. When a customer arrives, they join the end of the queue and wait until it is their turn to be served. Once the server is available, the customer is served and subsequently exits the system.

The objectives of the experiment are as follows:

1. **Simulation Execution**

   According to the original problem, and using the simulation objectives and program from the textbook, run the simulation and calculate the expected average delay in queue of the n customers completing their delays during the simulation, the expected average number of customers in the queue, and the expected utilization of the server.

2. **Random Seed Variation**

   Change the seed of the random number generator and run the simulation program 10 times. Discuss the variation in the outputs across these runs.

3. **Performance Measures Modification**

   Modify the code for the single-server queue to compute and report the following performance measures:

   a) The time-average number of customers in the system
   b) The average total time customers spend in the system
   c) The maximum queue length observed
   d) The maximum delay experienced by a customer in the queue
   e) The maximum time a customer spends in the system
   f) The proportion of customers who experience a delay in the queue exceeding 1 minute

   After modification, run the program and collect the simulation output.

4. **Operational Time Adjustment**

   Assume the facility operates from 9 A.M. to 5 P.M. (time 0 to 480 minutes), and continues to serve all customers present at closing time. Modify the code to reflect this stopping rule, then estimate and report the same performance measures as in step 3.

5. **Queue Capacity Limitation**

   Introduce a queue capacity limit of two customers. If a customer arrives to find

the queue full, they leave without entering the queue (balking). Simulate this system with a fixed operation time of 480 minutes. Estimate and report the same performance measures as in step 3, as well as the expected number of customers who balk.

6. **Integrated Visual Program**

Modify the original C program to develop an integrated visual program for running the simulation experiments of above.

Methods for collecting and computing the additional performance measures

The objective of this experiment is to simulate a single-server queue (M/M/1 queue) using the C++ programming language with the Microsoft Foundation Class (MFC) library. We aim to investigate various performance measures and system behaviors under different conditions and configurations. The experiment is divided into six parts: running the basic simulation, varying random seeds, modifying performance measures, adjusting operational times, introducing queue capacity limits, and developing an integrated visual program.

1. Part 1: Simulation Execution

Problem Analysis:

We aim to simulate a single-server queue with exponential inter-arrival and service times, calculate the average delay in the queue, the average number of customers in the queue, and the server utilization.

Solution Approach:

a) Implement the M/M/1 queue simulation in C++ using MFC.
b) Generate exponential inter-arrival and service times using a random number generator.
c) Track the arrival, service start, and departure times of each customer.
d) Compute the average delay in the queue, the average number of customers in the queue, and server utilization.

Code Implementation:

```
1.  void mm1::report(void) /* Report generator function. */
2.  {
3.      /* Compute and write estimates of desired measures of performance
    . */
4.      results.push_back(total_of_delays / num_custs_delayed);
5.      results.push_back(area_num_in_q / sim_time);
6.      results.push_back(area_server_status / sim_time);
7.      results.push_back(sim_time);
```

```
8.
9.      results.push_back(total_num_in_system  / sim_time); // 计算系统中的
    平均人数
10.     results.push_back(total_time_in_system  / num_custs_delayed); // 计
    算顾客在系统中的平均总时间
11.     results.push_back(max_queue_length); // 记录最大队列长度
12.     results.push_back(max_delay);     // 将最大延迟写入结果向量
13.     results.push_back(max_time_in_system); // 写入最大系统时间
14.
15.     float proportion_delayed_over_1_min = (float)num_custs_delayed_ov
    er_1_min / num_custs_delayed;
16.     results.push_back(proportion_delayed_over_1_min);
17. }
```

2. Part 2: Random Seed Variation

Problem Analysis:

We aim to assess the variability in the simulation outputs by running the simulation multiple times with different random seeds.

Solution Approach:

a) Modify the random number generator seed before each simulation run.
b) Execute the simulation 10 times with different seeds.
c) Collect and analyze the results for variability.

Code Implementation:

```
1.  void Queue::OnBnClickedButton1()
2.  {
3.      // 获取编辑控件的样式
4.      LONG style = GetWindowLong(seed_by_index.GetSafeHwnd(), GWL_STYLE
    );
5.
6.      long seed_long = 0;
7.      CWnd* pEditCtrl;
8.
9.      // 检查是否包含 ES_READONLY 样式
10.     if (style & ES_READONLY)
11.     {
12.         // 获取种子值
13.         GetDlgItemText(IDC_EDIT7, seed);
14.         pEditCtrl = GetDlgItem(IDC_EDIT7); // 获取编辑框的指针
15.         seed_long = _wtol(seed);
16.
17.         // 更改种子值
18.         if (pEditCtrl != nullptr && seed_long > 0) {
```

```
            lcgrandst(seed_long, 1);
        }
    }
    else
    {
        CString index;
        GetDlgItemText(IDC_EDIT12, index);

        pEditCtrl = GetDlgItem(IDC_EDIT12);  // 获取编辑框的指针
        int index_num = _wtol(index);

        // 更改种子值
        if (pEditCtrl != nullptr && index_num > 0 && index_num < getl
ength()) {
            resetStream();
            seed_long = lcgrandgt(index_num);
            seed.Format(_T("%ld"), seed_long);
            lcgrandst(seed_long, 1);
        }
        else {
            AfxMessageBox(_T("The index is not valid."));
        }
    }
    previousSeed = seed_long;
}
```

3. Part 3: Performance Measures Modification

Problem Analysis:

We aim to enhance the simulation to compute additional performance measures.

Solution Approach:

a) Add variables to track new performance metrics.
b) Modify the simulation loop to update these variables.
c) Compute and report the required metrics.

Modified Code Implementation:

a) MM1.h

```
#mm1.h
#pragma once

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <iostream>
#include <vector>
```

```
9.
10. using namespace std;
11.
12. class mm1 {
13. private:
14.     #define Q_LIMIT 100 /* Limit on queue length. */
15.     #define BUSY 1      /* Mnemonics for server's being busy */
16.     #define IDLE 0      /* and idle. */
17.
18.     int next_event_type, num_custs_delayed, num_delays_required, num_
    events,
19.         num_in_q, server_status;
20.     float area_num_in_q, area_server_status, mean_interarrival, mean_
    service,
21.         sim_time, time_arrival[Q_LIMIT + 1], time_last_event, time_ne
    xt_event[3],
22.         total_of_delays;
23.
24.     vector<float> results; // 用于存储结果
25.
26.     float total_num_in_system;  // 用于累积系统中的人数
27.     float total_time_in_system;  // 用于累积系统中的总时间
28.     int max_queue_length;  // 用于记录最大队列长度
29.     float max_delay = 0.0; // 在全局定义中添加最大延迟变量
30.     float max_time_in_system;  // 最大系统时间
31.     float delay_excess;  // 用于记录延迟超过的时间
32.
33.     /* Initialize the count for customers delayed over 1 minute. */
34.     int num_custs_delayed_over_1_min = 0;
35.
36. public:
37.     vector<float> mm1function(float mean_interarrival, float mean_ser
    vice, int num_delays_required, float delay_excess);
38.     void initialize();
39.     int timing();
40.     int arrive();
41.     void depart();
42.     void report();
43.     void update_time_avg_stats();
44.     float expon(float mean);
45. };
```

b) mm1.cpp

```
1. void mm1::report(void) /* Report generator function. */
```

```
{
    /* Compute and write estimates of desired measures of performance
. */
    //results.push_back(mean_interarrival);
    //results.push_back(mean_service);
    //results.push_back(num_delays_required);

    results.push_back(total_of_delays / num_custs_delayed);
    results.push_back(area_num_in_q / sim_time);
    results.push_back(area_server_status / sim_time);
    results.push_back(sim_time);

    results.push_back(total_num_in_system  / sim_time); // 计算系统中的
平均人数
    results.push_back(total_time_in_system  / num_custs_delayed);  // 计
算顾客在系统中的平均总时间
    results.push_back(max_queue_length);  // 记录最大队列长度
    results.push_back(max_delay);     // 将最大延迟写入结果向量
    results.push_back(max_time_in_system);  // 写入最大系统时间

    float proportion_delayed_over_1_min = (float)num_custs_delayed_ov
er_1_min / num_custs_delayed;
    results.push_back(proportion_delayed_over_1_min);
}

void mm1::update_time_avg_stats(void) /* Update area accumulators for
 time-average
                                      statistics. */
{
    float time_since_last_event;

    /* Compute time since last event, and update last-event-
time marker. */

    time_since_last_event = sim_time - time_last_event;
    time_last_event = sim_time;

    /* Update area under number-in-queue function. */

    area_num_in_q += num_in_q * time_since_last_event;

    /* Update area under server-busy indicator function. */

    area_server_status += server_status * time_since_last_event;
```

```
40.
41.     // 更新系统中的人数
42.     total_num_in_system += (num_in_q + server_status) * time_since_la
    st_event;
43.     // 更新系统中的总时间
44.     total_time_in_system += (num_in_q + server_status) * time_since_l
    ast_event;
45. }
```

4. Part 4: Operational Time Adjustment

Problem Analysis:

Adjust the simulation to reflect a fixed operational time from 9 A.M. to 5 P.M. and serve all remaining customers at closing time.

Solution Approach:

a) Modify the simulation to track operational time.

b) Continue serving customers who are in the queue at closing time.

Modified Code Implementation:

```
1.  vector<float> mm1alt::mm1Alt(float m_l, float m_s, float t_e, float d
    _e) /* Main function. */
2.  {
3.      /* Specify the number of events for the timing function. */
4.
5.      num_events = 3;
6.
7.      /* Read input parameters. */
8.
9.      mean_interarrival = m_l;
10.     mean_service = m_s;
11.     time_end = t_e;
12.     delay_excess = d_e;
13.
14.     /* Initialize the simulation. */
15.
16.     initialize();
17.
18.     /* Run the simulation until it terminates after an end-
    simulation event
19.        (type 3) occurs. */
20.
21.     do
22.     {
23.         /* Determine the next event. */
```

```
24.
25.         if (timing() == 1) {
26.             return results;
27.         }
28.
29.         /* Update time-average statistical accumulators. */
30.
31.         update_time_avg_stats();
32.
33.         /* Invoke the appropriate event function. */
34.
35.         switch (next_event_type)
36.         {
37.         case 1:
38.             if (arrive() == 2) {
39.                 return results;
40.             }
41.             break;
42.         case 2:
43.             depart();
44.             break;
45.         case 3:
46.             report();
47.             break;
48.         }
49.         /* If the event just executed was not the end-
    simulation event (type 3),
50.            continue simulating.  Otherwise, end the simulation. */
51.     } while (next_event_type != 3);
52.
53.     return results;
54. }
```

5. Part 5: Queue Capacity Limitation

Problem Analysis:

Introduce a queue capacity limit and simulate balking behavior.

Solution Approach:

a) Check queue length before admitting new customers.
b) Implement balking logic when the queue is full.

Modified Code Implementation:

```
1.  int balk::arrive(void) /* Arrival event function. */
2.  {
```

```
3.      float delay;
4.
5.      /* Schedule next arrival. */
6.
7.      time_next_event[1] = sim_time + expon(mean_interarrival);
8.
9.      /* Check to see whether server is busy. */
10.
11.     if (server_status == BUSY)
12.     {
13.         /* Server is busy, so increment number of customers in queue.
    */
14.
15.         if (num_in_q < q_limit) {
16.             ++num_in_q;
17.         }
18.         else {
19.             ++num_custs_balked;
20.             return 0;
21.         }
22.
23.         /* Update max_queue_length if needed. */
24.
25.         if (num_in_q > max_queue_length) {
26.             max_queue_length = num_in_q;
27.         }
28.
29.         /* Check to see whether an overflow condition exists. */
30.
31.         if (num_in_q > Q_LIMIT)
32.         {
33.             /* The queue has overflowed, so stop the simulation. */
34.
35.             printf("\nOverflow of the array time_arrival at");
36.             printf(" time %f", sim_time);
37.             results.push_back(sim_time);
38.             return 2;
39.         }
40.
41.         /* There is still room in the queue, so store the time of arr
    ival of the
42.            arriving customer at the (new) end of time_arrival. */
43.
44.         time_arrival[num_in_q] = sim_time;
```

```
    } else {
        /* Server is idle, so arriving customer has a delay of zero. (The
           following two statements are for program clarity and do not affect
           the results of the simulation.) */

        delay = 0.0;
        total_of_delays += delay;

        /* Increment the number of customers delayed, and make server busy. */

        ++num_custs_delayed;
        server_status = BUSY;

        // 在顾客开始服务时计算延迟并更新最大延迟变量
        delay = sim_time - time_arrival[1];
        if (delay > max_delay) {
            max_delay = delay;
        }

        // 如果顾客延迟时间超过一分钟
        if (delay > delay_excess) {
            ++num_custs_delayed_over_1_min;
        }
        delay = 0.0;

        /* Schedule a departure (service completion). */

        time_next_event[2] = sim_time + expon(mean_service);
    }
    return 0;
}
```

6. Part 6: Integrated Visual Program

Problem Analysis:

Develop an integrated visual program using MFC to facilitate running simulation experiments.

Solution Approach:

a) Use MFC to create a GUI with controls for setting simulation parameters. Display real-time simulation results in the GUI.

b) Implement buttons for running different simulation scenarios.

Code Implementation:

```
1.  void Queue::OnBnClickedButton6()
2.  {
3.      // 暂存上一次的按钮 ID
4.      int tmp = m_nSelectRadio;
5.
6.      // 定义一个存储单选按钮 ID 的数组
7.      int radioButtons[] = { IDC_RADIO1, IDC_RADIO2, IDC_RADIO3, IDC_RA
    DIO4 };
8.
9.      // 定义一个存储单选按钮状态的数组
10.     int radioStates[4];
11.
12.     // 使用循环遍历单选按钮数组
13.     for (int i = 0; i < 4; ++i) {
14.         // 获取当前单选按钮的状态
15.         radioStates[i] = ((CButton*)GetDlgItem(radioButtons[i]))->Get
    Check();
16.         if (radioStates[i] == BST_CHECKED) {
17.             m_nSelectRadio = radioButtons[i];
18.         }
19.     }
20.
21.     // 判断是否需要更新列表的列
22.     if (tmp != m_nSelectRadio) {
23.         // 初始化控件状态
24.         ClearControls(this);
25.         UpdateColumn(this);
26.     }
27.
28.     // 获取编辑框中的值
29.     CString mean_interarrival;
30.     GetDlgItemText(IDC_EDIT1, mean_interarrival);
31.     CString mean_service;
32.     GetDlgItemText(IDC_EDIT2, mean_service);
33.     CString replication;
34.     GetDlgItemText(IDC_EDIT6, replication);
35.     CString delay;
36.     GetDlgItemText(IDC_EDIT5, delay);
37.
38.     // 获取编辑控件的指针
39.     CString strTemp;
40.     output.GetWindowText(strTemp);
```

```
41.     if (!strTemp.IsEmpty()) {
42.         // 追加新的文本
43.         strTemp += _T("\r\n") + seed;
44.     }
45.     else {
46.         strTemp = seed;
47.     }
48.
49.     // 将新的文本设置到文本框中
50.     output.SetWindowText(strTemp);
51.
52.     // 将 CString 转换为 float
53.     int replication_int = _ttoi(replication);
54.     float mean_interarrival_float = _ttof(mean_interarrival);
55.     float mean_service_float = _ttof(mean_service);
56.     float delay_excess = _ttof(delay);
57.
58.     // 更新延迟列的标题
59.     UpdateColumnDelay(this, delay_excess);
60.
61.     // 获取列表控件的行数
62.     rowCount = list.GetItemCount();
63.
64.     // 判断哪一个单选按钮是选中的
65.     switch (m_nSelectRadio) {
66.     case IDC_RADIO1: // Fixed Customers
67.     {
68.         CString num_delays_required;
69.         GetDlgItemText(IDC_EDIT3, num_delays_required);
70.         int num_delays_required_float = _ttoi(num_delays_required);
71.
72.         mm1 m;
73.         for (int i = rowCount; i < replication_int + rowCount; i++)
74.         {
75.             vector<float> results = m.mm1function(mean_interarrival_f
    loat, mean_service_float, num_delays_required_float, delay_excess);
76.             ShowResultsInListCtrl(list, results, i);
77.         }
78.         break;
79.     }
80.     case IDC_RADIO2: // Fixed Time
81.     {
82.         // 获取编辑框中的值
83.         CString open_time_value;
```

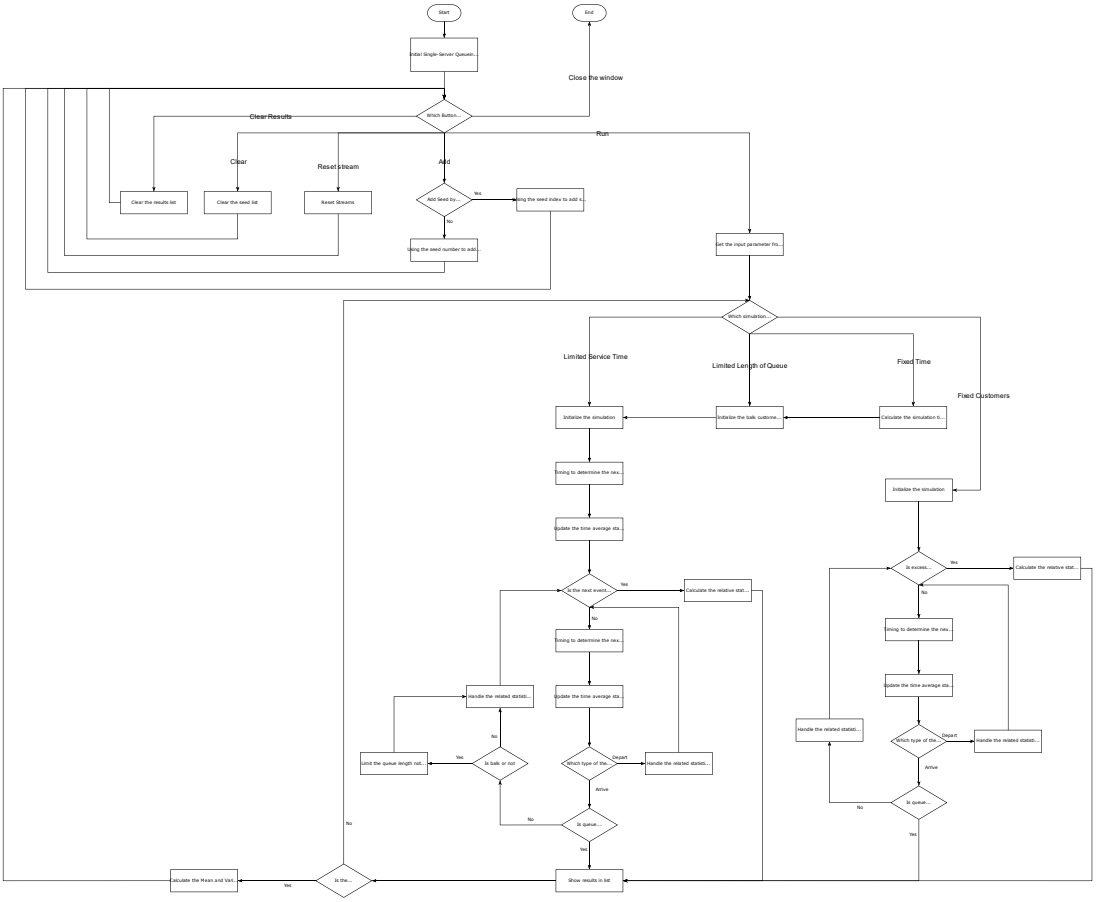
```
84.         GetDlgItemText(IDC_EDIT10, open_time_value);
85.         CString close_time_value;
86.         GetDlgItemText(IDC_EDIT11, close_time_value);
87.
88.         // 将 CString 转换为 float
89.         float o_t = _ttof(open_time_value);
90.         float c_t = _ttof(close_time_value);
91.         float time_simulation_ended = (c_t - o_t) * 60;
92.
93.         mm1alt m;
94.         for (int i = rowCount; i < replication_int + rowCount; i++)
95.         {
96.             vector<float> results = m.mm1Alt(mean_interarrival_float,
    mean_service_float, time_simulation_ended, delay_excess);
97.             ShowResultsInListCtrl(list, results, i);
98.         }
99.         break;
100.       }
101.       case IDC_RADIO3: // Limited Service Time
102.       {
103.           // 获取编辑框中的值
104.           CString time_simulation_ended;
105.           GetDlgItemText(IDC_EDIT4, time_simulation_ended);
106.
107.           // 将 CString 转换为 float
108.           float t_e = _ttof(time_simulation_ended);
109.
110.           mm1alt m;
111.           for (int i = rowCount; i < replication_int + rowCount; i++
    )
112.           {
113.               vector<float> results = m.mm1Alt(mean_interarrival_flo
    at, mean_service_float, t_e, delay_excess);
114.               ShowResultsInListCtrl(list, results, i);
115.           }
116.           break;
117.       }
118.       case IDC_RADIO4: // Limited Queue Length
119.       {
120.           // 获取编辑框中的值
121.           CString max_length;
122.           GetDlgItemText(IDC_EDIT10, max_length);
123.           CString open_time_value;
124.           GetDlgItemText(IDC_EDIT10, open_time_value);
```

```
            CString close_time_value;
            GetDlgItemText(IDC_EDIT11, close_time_value);

            // 将 CString 转换为 float
            float o_t = _ttof(open_time_value);
            float c_t = _ttof(close_time_value);
            int max_length_int = _ttoi(max_length);

            balk m;
            for (int i = rowCount; i < replication_int + rowCount; i++
    )
            {
                vector<float> results = m.mm1Balk(mean_interarrival_fl
    oat, mean_service_float, o_t, c_t, max_length_int, delay_excess);
                ShowResultsInListCtrl(list, results, i);
            }
            break;
        }
        }

        // 计算均值和方差
        vector<float> means = CalculateMean(allResults);
        vector<float> variances = CalculateVariance(allResults, means)
    ;

        // 插入均值行
        ShowSpecialResultsInListCtrl(list, means, _T("mean"));

        // 插入方差行
        ShowSpecialResultsInListCtrl(list, variances, _T("variance"));


    }
```

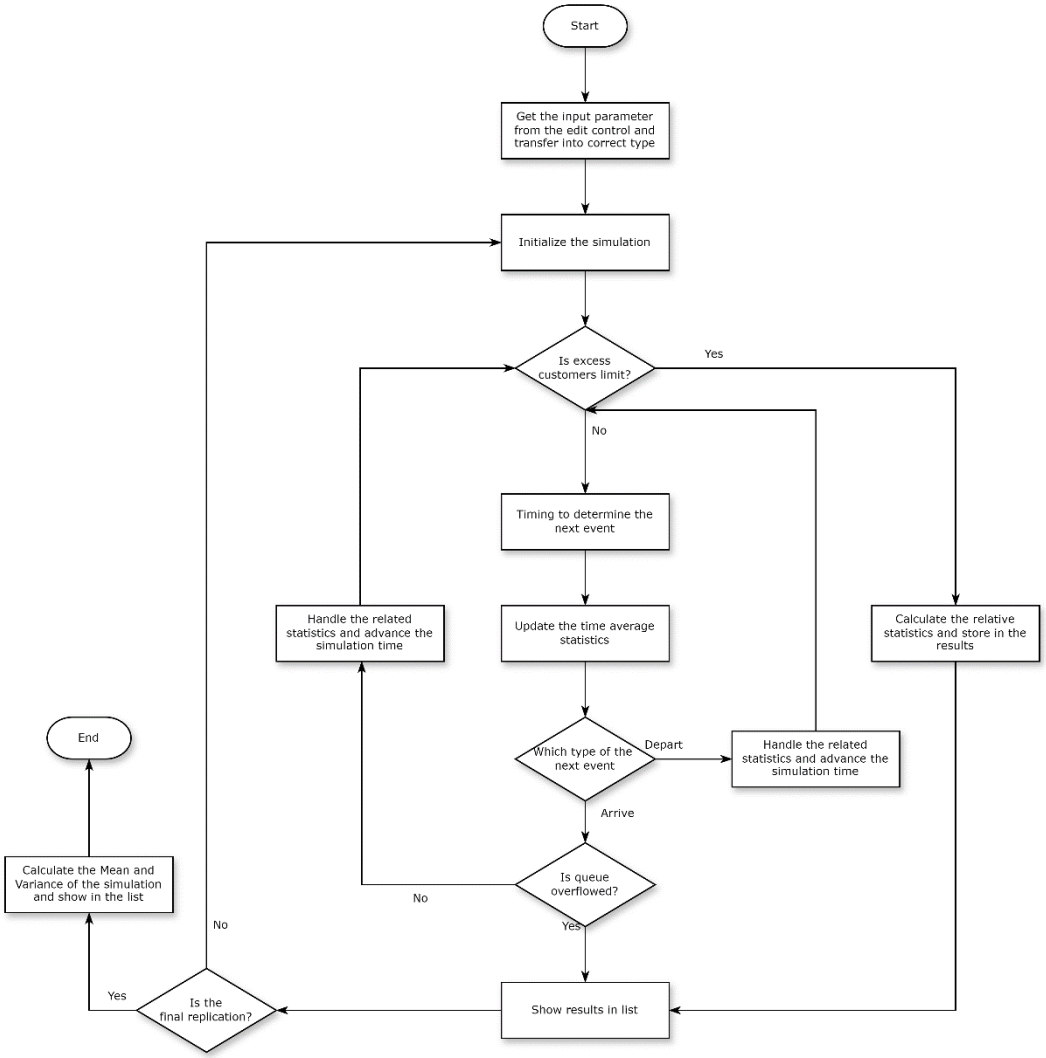
The series of simulations and modifications allow us to study the performance of an M/M/1 queue under various conditions. By using C++ and MFC, we have created a flexible and interactive tool to explore queue behavior, providing valuable insights into system performance and customer experiences.

Flowchart of modified program

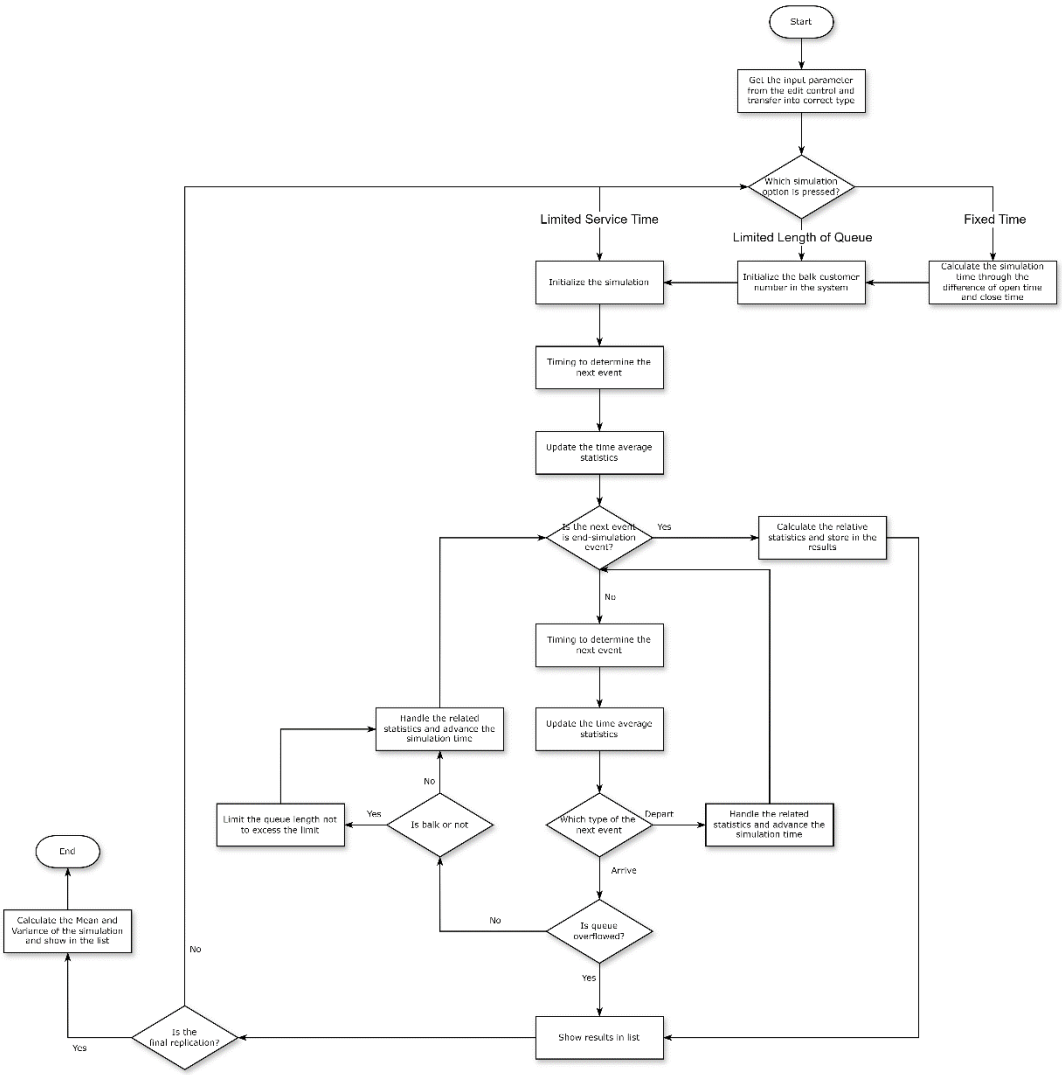
1. The flowchart of the whole single-server queueing system



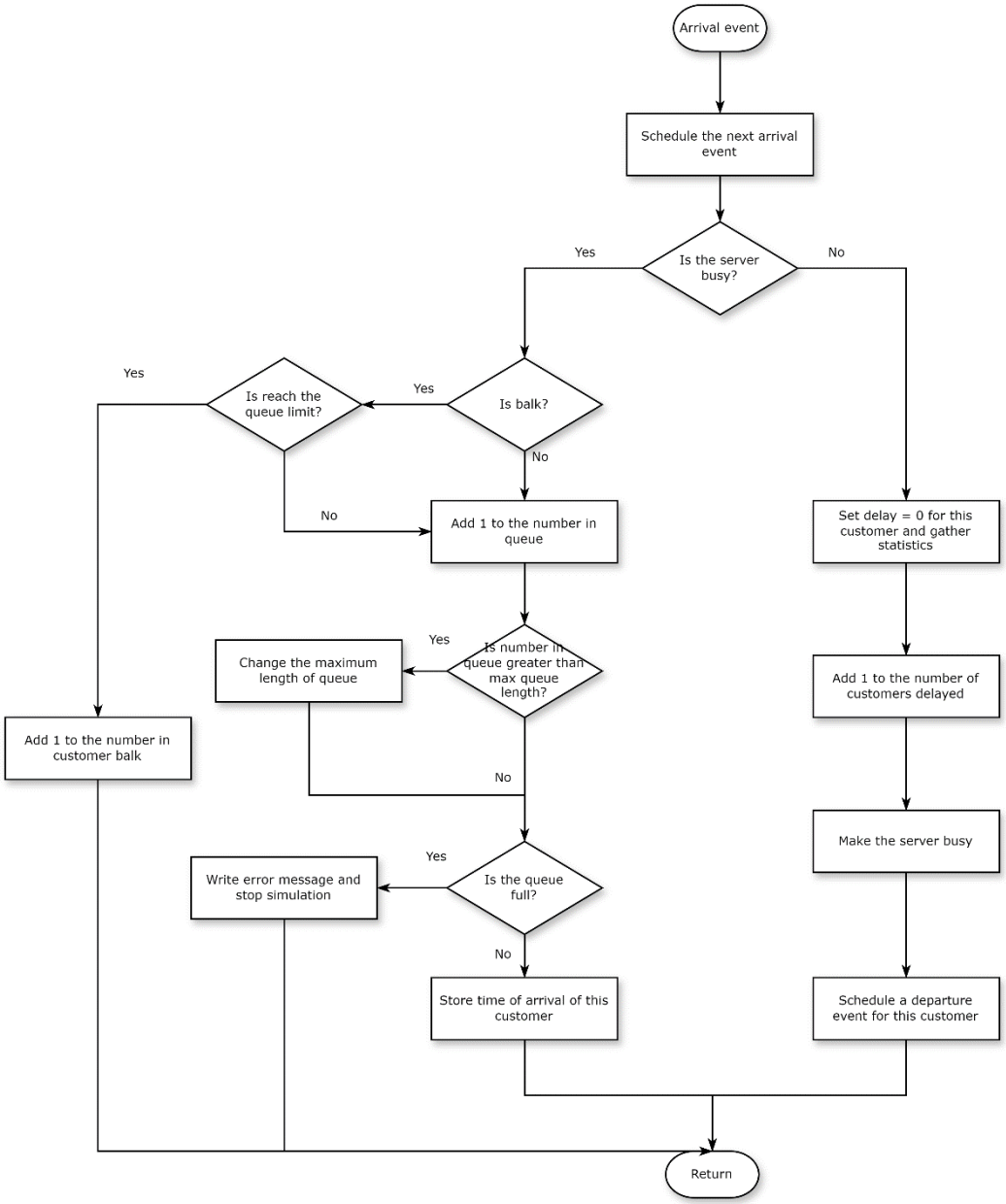
2. The flowchart of the original problem



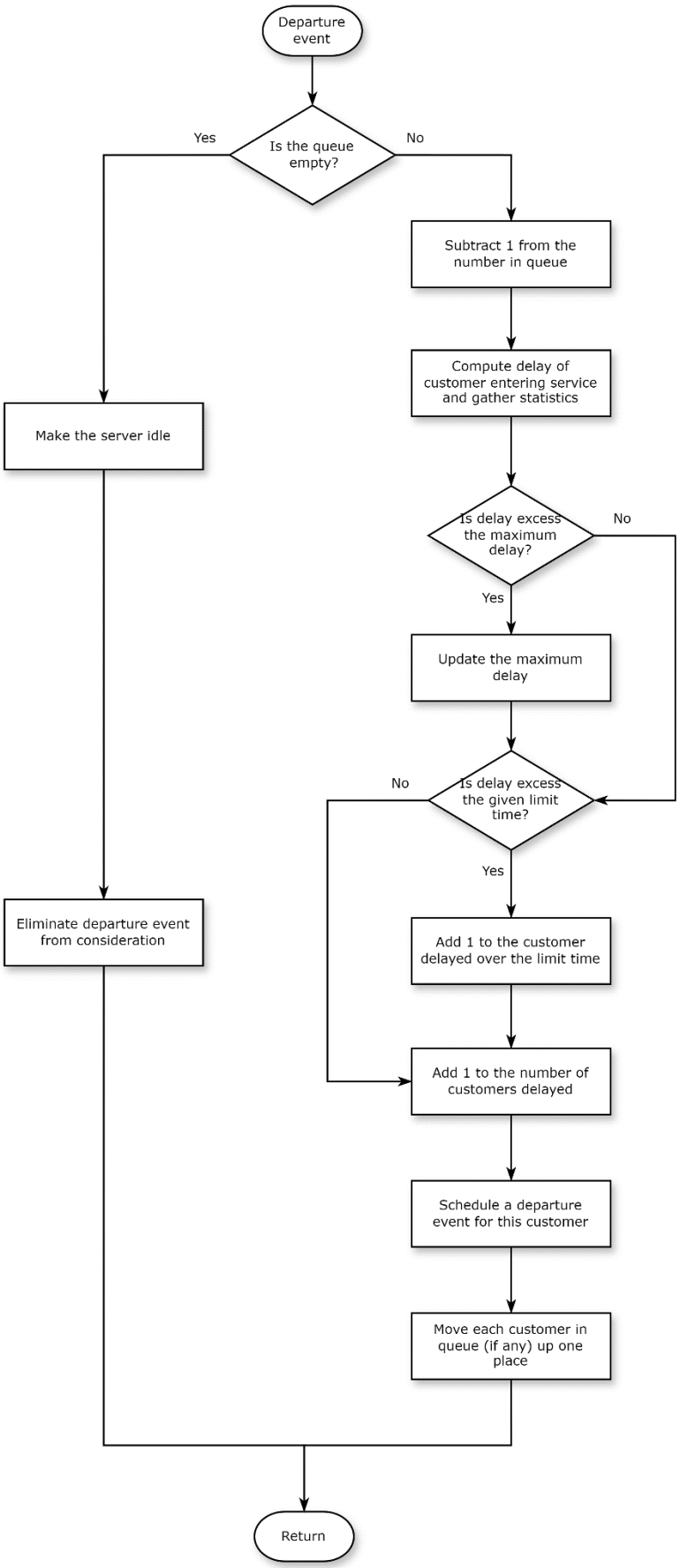
3. The flowchart of the modified program



4. The flowchart of the arrival routine of modified program



5. The flowchart of the departure routine of modified program


Departure event
Is the queue empty?
Yes
No
Make the server idle
Eliminate departure event from consideration
Subtract 1 from the number in queue
Compute delay of customer entering service and gather statistics
Is delay excess the maximum delay?
No
Yes
Update the maximum delay
Is delay excess the given limit time?
No
Yes
Add 1 to the customer delayed over the limit time
Add 1 to the number of customers delayed
Schedule a departure event for this customer
Move each customer in queue (if any) up one place
Return

Output of experiments

1. Fixed Customers condition with 10 replications in the same seed

Replicati	Average c	Average r	Service u	Time simulation enc	Time-aver	Average t	Maximum c	Maximum c	Maximum t	Proportic
1	0.429994	0.418316	0.460046	1027.914795	0.878362	0.902882	11	4.127502	7.827087	0.162
2	0.43685	0.437931	0.488528	997.5304565	0.926459	0.924172	9	4.805908	6.361115	0.165
3	0.49321	0.505649	0.507653	975.4005737	1.013302	0.988375	8	5.198914	5.482758	0.178
4	0.536269	0.53752	0.503709	997.6716309	1.041231	1.038806	15	7.348495	6.965637	0.176
5	0.416974	0.411736	0.491203	1012.722046	0.902939	0.914426	7	4.878998	6.877014	0.159
6	0.410879	0.415556	0.504068	990.3730469	0.919624	0.910771	7	3.223694	6.585083	0.159
7	0.515332	0.520427	0.506377	990.2085571	1.026804	1.01675	7	4.943542	5.567696	0.203
8	0.393587	0.377689	0.465096	1042.093872	0.842785	0.878261	8	4.078125	6.312012	0.138
9	0.436366	0.426164	0.487356	1023.938477	0.91352	0.935388	7	4.95871	5.271885	0.17
10	0.549085	0.588362	0.514189	940.9694214	1.102551	1.037467	12	5.673096	5.636414	0.197
Mean	0.461854	0.463935	0.492823	999.8822876	0.956758	0.95473	9.1	4.923698	6.28867	0.1707
Variance	0.002855	0.004252	0.0003	757.5703903	0.006269	0.003229	6.69	1.077341	0.58934	0.000327

2. Fixed Customers condition with 10 replications in different seed

Replicati	Average c	Average r	Service u	Time sim	Time-aver	Average t	Maximum c	Maximum c	Maximum t	Proportic
1	0.429994	0.418316	0.460046	1027.915	0.878362	0.902882	11	4.127502	7.827087	0.162
2	0.523814	0.55194	0.532704	949.0419	1.084644	1.029372	7	4.489319	5.251038	0.202
3	0.548567	0.54495	0.527606	1006.637	1.072556	1.079675	7	4.755402	6.581299	0.213
4	0.462772	0.463726	0.484129	997.942	0.947855	0.945904	8	4.190933	8.978088	0.191
5	0.522858	0.539171	0.509975	969.7444	1.049145	1.017403	8	4.28302	6.729772	0.196
6	0.396091	0.394701	0.477623	1003.521	0.872324	0.875396	6	3.691833	6.137409	0.154
7	0.552004	0.579421	0.522502	952.8242	1.101924	1.04994	8	4.609802	5.194702	0.209
8	0.465611	0.442949	0.490942	1051.163	0.933891	0.981672	7	4.080688	6.076097	0.177
9	0.458276	0.462388	0.491076	991.1072	0.953464	0.944985	8	4.681335	6.958664	0.175
10	0.414864	0.421145	0.490022	985.0843	0.911167	0.897577	9	4.539185	7.040253	0.145
Mean	0.477485	0.481871	0.498662	993.4981	0.980533	0.972481	7.9	4.344902	6.677441	0.1824
Variance	0.002834	0.003924	0.000503	907.4104	0.006969	0.004417	1.69	0.098158	1.172376	0.000499

3. Limited Service Time condition (480) with 10 replications in the same seed

Replicati	Average c	Average r	Service u	Time sim	Time-aver	Average t	Maximum c	Maximum c	Maximum t	Proportic
1	0.398633	0.394481	0.464231	475	0.858712	0.867751	6	3.440208	6.863861	0.149474
2	0.730239	0.784396	0.554328	506	1.338724	1.269936	11	5.749157	7.106548	0.245059
3	0.495427	0.472232	0.479668	457	0.9519	0.999807	8	4.94281	5.7323	0.205689
4	0.336714	0.331102	0.452763	472	0.783866	0.797152	6	3.403412	6.046997	0.125
5	0.474167	0.443543	0.436162	449	0.879705	0.940442	7	6.020905	6.106445	0.171492
6	0.448689	0.46458	0.487291	497	0.951871	0.919312	6	5.198853	5.482758	0.158954
7	0.618964	0.613806	0.47591	476	1.089717	1.098874	15	7.348511	6.965637	0.197479
8	0.456763	0.462473	0.508062	486	0.970535	0.958553	8	3.995728	6.933548	0.162551
9	0.473454	0.468522	0.530974	475	0.999495	1.010016	9	4.720917	5.430969	0.166316
10	0.443183	0.440413	0.495857	477	0.93627	0.942159	6	4.878937	6.876999	0.171908
Mean	0.487623	0.487555	0.488525	477	0.97608	0.9804	8.2	4.969944	6.354606	0.175392
Variance	0.011169	0.014311	0.001142	254	0.020777	0.015266	7.56	1.323591	0.396288	0.001002

4. Limited Service Time condition (480) with 10 replications in different seed

Replicati	Average c	Average r	Service u	Time sim	Time-aver	Average t	Maximum c	Maximum c	Maximum t	Proportic
1	0.398633	0.394481	0.464231	475	0.858712	0.867751	6	3.440208	6.863861	0.149474
2	0.539831	0.561199	0.533723	499	1.094923	1.053232	7	3.829834	5.251038	0.206413
3	0.619806	0.619806	0.538616	480	1.158422	1.158422	7	4.755402	6.581299	0.25
4	0.477556	0.49248	0.510208	495	1.002688	0.972303	7	4.190933	4.571716	0.210101
5	0.473125	0.469182	0.491886	476	0.961067	0.969144	6	3.87854	6.729772	0.186975
6	0.480621	0.51867	0.534395	518	1.053065	0.975813	6	3.069427	6.137409	0.214286
7	0.568676	0.585262	0.523746	494	1.109009	1.077579	8	4.155701	5.00351	0.232794
8	0.371372	0.348162	0.469256	450	0.817418	0.871912	5	4.050117	6.076097	0.142222
9	0.441507	0.442427	0.476645	481	0.919071	0.917161	6	4.262573	6.958664	0.178794
10	0.334316	0.330137	0.465796	474	0.795932	0.806008	5	4.539185	7.040253	0.124473
Mean	0.470544	0.47618	0.50085	484.2	0.977031	0.966932	6.3	4.017192	6.121362	0.189553
Variance	0.007127	0.008741	0.000848	302.76	0.014655	0.010371	0.81	0.221359	0.708921	0.001505

5. Fixed Time (9 to 17) condition with 10 replications in the same seed

Replicati	Average c	Average r	Service u	Time sim	Time-avei	Average t	Maximum c	Maximum c	Maximum t	Proportic
1	0.398633	0.394481	0.464231	475	0.858712	0.867751	6	3.440208	6.863861	0.149474
2	0.730239	0.784396	0.554328	506	1.338724	1.269936	11	5.749157	7.106548	0.245059
3	0.495427	0.472232	0.479668	457	0.9519	0.999807	8	4.94281	5.7323	0.205689
4	0.336714	0.331102	0.452763	472	0.783866	0.797152	6	3.403412	6.046997	0.125
5	0.474167	0.443543	0.436162	449	0.879705	0.940442	7	6.020905	6.106445	0.171492
6	0.448689	0.46458	0.487291	497	0.951871	0.919312	6	5.198853	5.482758	0.158954
7	0.618964	0.613806	0.47591	476	1.089717	1.098874	15	7.348511	6.965637	0.197479
8	0.456763	0.462473	0.508062	486	0.970535	0.958553	8	3.995728	6.933548	0.162551
9	0.473454	0.468522	0.530974	475	0.999495	1.010016	9	4.720917	5.430969	0.166316
10	0.443183	0.440413	0.495857	477	0.93627	0.942159	6	4.878937	6.876999	0.171908
Mean	0.487623	0.487555	0.488525	477	0.97608	0.9804	8.2	4.969944	6.354606	0.175392
Variance	0.011169	0.014311	0.001142	254	0.020777	0.015266	7.56	1.323591	0.396288	0.001002

6. Fixed Time (9 to 17) condition with 10 replications in different seed

Replicati	Average c	Average r	Service u	Time sim	Time-avei	Average t	Maximum c	Maximum c	Maximum t	Proportic
1	0.398633	0.394481	0.464231	475	0.858712	0.867751	6	3.440208	6.863861	0.149474
2	0.539831	0.561199	0.533723	499	1.094923	1.053232	7	3.829834	5.251038	0.206413
3	0.619806	0.619806	0.538616	480	1.158422	1.158422	7	4.755402	6.581299	0.25
4	0.477556	0.49248	0.510208	495	1.002688	0.972303	7	4.190933	4.571716	0.210101
5	0.473125	0.469182	0.491886	476	0.961067	0.969144	6	3.87854	6.729772	0.186975
6	0.480621	0.51867	0.534395	518	1.053065	0.975813	6	3.069427	6.137409	0.214286
7	0.568676	0.585262	0.523746	494	1.109009	1.077579	8	4.155701	5.00351	0.232794
8	0.371372	0.348162	0.469256	450	0.817418	0.871912	5	4.050117	6.076097	0.142222
9	0.441507	0.442427	0.476645	481	0.919071	0.917161	6	4.262573	6.958664	0.178794
10	0.334316	0.330137	0.465796	474	0.795932	0.806008	5	4.539185	7.040253	0.124473
Mean	0.470544	0.47618	0.50085	484.2	0.977031	0.966932	6.3	4.017192	6.121362	0.189553
Variance	0.007127	0.008741	0.000848	302.76	0.014655	0.010371	0.81	0.221359	0.708921	0.001505

7. Limited Length of Queue (9 to 17, Maximum length of queue is 2) condition with 10 replications in the same seed

Replicat	Average c	Average r	Service u	Time sim	Time-avei	Average t	Maximum c	Maximum c	Maximum t	Proportion of cust	Balk of c
1	0.287466	0.267702	0.441825	447	0.709528	0.761909	2	3.876495	5.59375	0.089485459	32
2	0.285938	0.268066	0.466142	450	0.734208	0.783155	2	2.683197	7.170258	0.079999998	29
3	0.278869	0.255347	0.465467	437	0.720814	0.791741	2	2.756989	5.931366	0.107551485	33
4	0.245691	0.240573	0.476237	470	0.71681	0.732061	2	2.613495	5.155182	0.078723401	28
5	0.273515	0.256421	0.46854	450	0.72496	0.773291	2	3.796844	5.158264	0.095555559	25
6	0.259376	0.248028	0.460605	459	0.708633	0.741054	2	2.447601	5.857544	0.078431375	36
7	0.253464	0.239207	0.444843	453	0.68405	0.724821	2	3.25238	5.283691	0.079470202	41
8	0.265176	0.239211	0.458549	433	0.697759	0.773498	2	3.266663	6.933548	0.085450344	29
9	0.276864	0.263597	0.480524	457	0.744121	0.781571	2	3.793915	4.738831	0.089715533	30
10	0.249138	0.224743	0.46252	433	0.687263	0.761861	2	3.055466	6.913391	0.080831409	27
Mean	0.26755	0.250289	0.462525	448.9	0.712815	0.762496	2	3.154305	5.873582	0.086521477	31
Variance	0.000207	0.000186	0.000133	128.29	0.00034	0.000469	0	0.254724	0.663141	7.94897E-05	20

8. Limited Length of Queue (9 to 17, Maximum length of queue is 2) condition with 10 replications in different seed

Replicati	Average c	Average r	Service u	Time sim	Time-avei	Average t	Maximum c	Maximum c	Maximum t	Proportic	Balk of c
1	0.287466	0.267702	0.441825	447	0.709528	0.761909	2	3.876495	5.59375	0.089485	32
2	0.310736	0.291962	0.484853	451	0.776815	0.826766	2	3.028076	6.539902	0.099778	42
3	0.288467	0.270808	0.465752	441	0.73656	0.801698	2	3.814117	6.278214	0.088435	30
4	0.268722	0.246328	0.426422	440	0.67275	0.733909	2	2.889465	4.657745	0.095455	32
5	0.246663	0.226622	0.435671	441	0.662294	0.720864	2	2.656158	6.729772	0.0839	28
6	0.30148	0.289547	0.45196	461	0.741507	0.772068	2	3.235909	6.137409	0.121475	44
7	0.295898	0.281729	0.486969	456	0.768698	0.809156	2	3.033569	5.160645	0.109649	39
8	0.282097	0.260939	0.48068	444	0.741619	0.801751	2	3.301801	4.945238	0.099099	20
9	0.257578	0.23826	0.428114	444	0.666373	0.720404	2	3.948792	6.958664	0.09009	17
10	0.228308	0.214514	0.439831	451	0.654345	0.69642	2	1.97229	7.040253	0.073171	20
Mean	0.276741	0.258841	0.454208	447.6	0.713049	0.764494	2	3.175667	6.004159	0.095054	30.4
Variance	0.00061	0.000637	0.000501	44.44	0.001918	0.001813	0	0.335793	0.670506	0.000165	80.04

Analysis of output

1. **Fixed Customers condition with 10 replications in the same seed and in different seed**

Summary of Results:

Fixed Customers, Single Seed:

- Average delay in queue: 0.4619
- Average number in queue: 0.4639
- Service utilization: 0.4928
- Time simulation ended: 999.88
- Time-average number in system: 0.9568
- Average total time in system of customers: 0.9547
- Maximum queue length: 9.1
- Maximum delay in queue: 4.924
- Maximum time in system: 6.289
- Proportion of customers delayed in queue in excess of 1 minute: 0.1707

Fixed Customers, Different Seeds:

- Average delay in queue: 0.4775
- Average number in queue: 0.4819
- Service utilization: 0.4987
- Time simulation ended: 993.50
- Time-average number in system: 0.9805
- Average total time in system of customers: 0.9725
- Maximum queue length: 7.9
- Maximum delay in queue: 4.345
- Maximum time in system: 6.677
- Proportion of customers delayed in queue in excess of 1 minute: 0.1824

Analysis

1. Simulation Execution

The simulation results show relatively consistent performance measures when using a fixed seed across multiple runs. The average delay in queue, average number in queue, and service utilization all exhibit minimal variance, indicating a stable and predictable queueing process under the given conditions.

2. Random Seed Variation

When the seed of the random number generator is varied across 10 runs, the results show some variation but remain within a similar range:

- Average delay in queue ranges from 0.3961 to 0.5520 with a mean of 0.4775.
- Average number in queue ranges from 0.3947 to 0.5794 with a mean of 0.4819.
- Service utilization ranges from 0.4600 to 0.5327 with a mean of 0.4987.
- Time simulation ended ranges from 949.04 to 1051.16 minutes with a mean of 993.50 minutes.
- Time-average number in system ranges from 0.8723 to 1.1019 with a mean of 0.9805.

- Average total time in system of customers ranges from 0.8754 to 1.0797 with a mean of 0.9725.
- Maximum queue length ranges from 6 to 11 with a mean of 7.9.
- Maximum delay in queue ranges from 3.6918 to 4.7554 with a mean of 4.3449.
- Maximum time in system ranges from 5.1947 to 8.9781 with a mean of 6.6774.
- Proportion of customers delayed in queue in excess of 1 minute ranges from 0.1450 to 0.2130 with a mean of 0.1824.

The variance observed in these metrics reflects the inherent randomness in the arrival and service processes, as well as the stochastic nature of queue lengths and delays. The variation is within acceptable limits, indicating that the system's performance is robust to changes in the random seed.

3. Performance Measures and Their Implications
The additional performance measures provide deeper insights into the system:
- Time-average number of customers in the system and average total time in the system are directly correlated with the average delay in queue, reflecting overall system congestion.
- Maximum queue length and maximum delay in queue highlight the worst-case scenarios that the system can handle, which are critical for stress testing and capacity planning.
- Proportion of customers delayed more than 1 minute is a significant metric for customer satisfaction and can guide service level agreements.

2. Limited Service Time condition (480) with 10 replications in the same seed and in different seed

The table below provides the results of running the simulation for the M/M/1 queue system under the condition of a limited service time of 480 minutes, using different random seeds for each run. Here is a detailed analysis of the observed performance measures:

Performance Measures and Their Variations

1. Average Delay in Queue:

- Mean: 0.4705 minutes
- Variance: 0.0071
- Observation: The average delay in the queue varied between 0.3343 minutes and 0.6198 minutes. This measure fluctuates moderately with different seeds, reflecting the inherent randomness in arrival and service processes.

2. Average Number in Queue:
- Mean: 0.4762 customers
- Variance: 0.0087
- Observation: The average number of customers in the queue ranged from 0.3301

to 0.6198. Similar to the average delay, this measure shows moderate variability, which indicates how congestion in the system changes with different random seeds.

3. Service Utilization:
- Mean: 0.5009
- Variance: 0.0008
- Observation: The server utilization fluctuated slightly around 50%, with the minimum utilization being 46.52% and the maximum being 53.86%. This small variance suggests that the server is consistently busy around half of the time, as expected in an M/M/1 system.

4. Time Simulation Ended:
- Mean: 484.2 minutes
- Variance: 302.76
- Observation: The simulation ended times varied between 450 minutes and 518 minutes. This wide range indicates significant variability in the number of customers processed before the facility closed.

5. Time-Average Number in System:
- Mean: 0.9770 customers
- Variance: 0.0147
- Observation: The average number of customers in the system (queue + being served) varied between 0.7959 and 1.1584. This measure shows the system's overall load and indicates the impact of both queue length and service time.

6. Average Total Time in the System:
- Mean: 0.9669 minutes
- Variance: 0.0104
- Observation: The average total time a customer spends in the system ranged from 0.8060 to 1.1584 minutes, which includes both waiting and service times. This variability reflects differences in customer experience based on queue length and service efficiency.

7. Maximum Queue Length:
- Mean: 6.3 customers
- Variance: 0.81
- Observation: The maximum queue length observed varied from 5 to 8 customers. This moderate variance indicates how peak congestion points can differ with each run.

8. Maximum Delay in Queue:
- Mean: 4.0172 minutes
- Variance: 0.2214

- Observation: The maximum delay experienced by a customer in the queue ranged from 3.0694 minutes to 4.7554 minutes. This measure indicates the worst-case waiting time scenario within the system for each run.

9. Maximum Time in the System:
- Mean: 6.1214 minutes
- Variance: 0.7089
- Observation: The maximum time a customer spent in the system varied between 4.5717 and 7.0403 minutes. This measure includes both waiting and service times and reflects the extreme cases of customer time in the system.

10. Proportion of Customers with Queue Delay Exceeding 1 Minute:
- Mean: 0.1896
- Variance: 0.0015
- Observation: The proportion of customers experiencing a queue delay exceeding 1 minute ranged from 0.1245 to 0.25. This metric indicates how often customers face significant delays, with a slight variation across runs.

3. Fixed Time (9 to 17) condition with 10 replications in the same seed and in different seed

The results from running the single-server queue simulation with a fixed operational time (9 AM to 5 PM) using different random seeds provide valuable insights into the system's performance under varying conditions. Here is an analysis based on the provided results:

Key Metrics

1. Average Delay in Queue:
- Mean: 0.4705 minutes
- Variance: 0.0071

The average delay in the queue across different seeds is around 0.47 minutes, indicating that customers typically wait for less than a minute before being served. The low variance suggests relatively stable queue delays across different simulation runs.

2. Average Number in Queue:
- Mean: 0.4762 customers
- Variance: 0.0087

On average, less than half a customer is waiting in the queue at any given time. This metric, along with the low variance, shows that the queue rarely builds up significantly.

3. Service Utilization:
- Mean: 0.5009
- Variance: 0.0008

The server is utilized approximately 50% of the time, indicating a balanced system where the server is neither idle too often nor overburdened. The low variance confirms consistent utilization across different runs.

4. Time Simulation Ended:
- Mean: 484.2 minutes
- Variance: 302.76

Simulations tend to end slightly after the 480-minute mark, which accounts for serving any customers still in the queue at closing time. The higher variance here reflects the variability in the number of customers arriving towards the end of the operational period.

5. Time-Average Number in System:
- Mean: 0.9770 customers
- Variance: 0.0147

On average, nearly one customer is present in the system (either being served or waiting) at any time. The low variance indicates a consistent load on the system.

6. Average Total Time in System:
- Mean: 0.9669 minutes
- Variance: 0.0104

Customers spend nearly a minute in the system from arrival to departure. This metric includes both waiting and service times, with a low variance indicating stable total time across runs.

7. Maximum Queue Length:
- Mean: 6.3 customers
- Variance: 0.81

The maximum queue length observed averages around 6.3 customers, suggesting occasional peaks in demand. The moderate variance indicates some variability in peak queue lengths.

8. Maximum Delay in Queue:
- Mean: 4.0172 minutes
- Variance: 0.2214

The maximum delay experienced by a customer in the queue averages around 4 minutes, with some variability as indicated by the variance. This metric highlights the worst-case waiting time scenarios.

9. Maximum Time in System:
- Mean: 6.1214 minutes
- Variance: 0.7089

The maximum time a customer spends in the system averages around 6 minutes, again indicating worst-case scenarios but with moderate variability.

10. Proportion of Customers Delayed in Queue Exceeding 1 Minute:
- Mean: 0.1896 (18.96%)
- Variance: 0.0015

Approximately 19% of customers experience a delay in the queue exceeding 1 minute. The low variance suggests consistent proportions across different runs.

Discussion on Variation Across Runs

The simulation outputs across different seeds show some variation in the key metrics, primarily due to the stochastic nature of arrival and service processes. However, the means and variances indicate the following:

- Stability: Metrics such as average delay in queue, average number in queue, service utilization, and time-average number in system exhibit low variances, suggesting stable system performance across different runs.
- Peaks in Demand: Metrics like maximum queue length and maximum delay in queue show higher variances, reflecting occasional but significant peaks in demand.
- Customer Experience: The proportion of customers experiencing delays over 1 minute remains relatively stable, indicating consistent customer experience across runs.

Overall, the analysis demonstrates that the single-server queue system operates efficiently with predictable performance, despite some variability in peak conditions.

4. Limited Length of Queue (9 to 17, Maximum length of queue is 2) condition with 10 replications in the same seed and in different seed

The simulation of a single-server queue with a FIFO discipline and a maximum queue length of 2 customers was conducted, varying the random seed for 10 runs. The objective was to observe the variability in key performance measures.

Analysis of Variation

1. Average Delay in Queue:
- The average delay in the queue ranges from 0.2283 to 0.3107 minutes.
- The variance is low (0.0006), indicating relatively stable average delays across different seeds.

2. Average Number in Queue:
- The average number in the queue varies from 0.2145 to 0.2920.
- The low variance (0.0006) suggests consistent queuing behavior.

3. Server Utilization:
- Server utilization ranges from 0.4264 to 0.4870.
- With a variance of 0.0005, the utilization rate is fairly stable across different runs.

4. Time Simulation Ended:
- The end time of the simulation varies slightly from 440 to 461 minutes.
- The variance (44.44) is moderate, showing some variation in the length of the simulation period.

5. Time-Average Number in System:
- The time-average number of customers in the system ranges from 0.6543 to 0.7768.
- The higher variance (0.0019) compared to the queue indicates more fluctuation in the total system occupancy.

6. Average Total Time in System:
- The average total time customers spend in the system varies from 0.6964 to 0.8268 minutes.
- Variance is 0.0018, reflecting some variability but overall stability.

7. Maximum Queue Length:
- The maximum queue length observed is consistently 2 across all runs, showing the impact of the queue capacity limit.

8. Maximum Delay in Queue:
- The maximum delay ranges from 1.9723 to 3.9488 minutes.
- The variance (0.3358) indicates noticeable variability in peak delays.

9. Maximum Time in System:
- The maximum time in the system ranges from 4.6577 to 7.0403 minutes.
- The variance (0.6705) shows significant differences in the longest times spent in the system.

10. Proportion of Delays Exceeding 1 Minute:
- The proportion of customers experiencing delays over 1 minute varies from 0.0732 to 0.1215.
- The variance (0.0002) suggests some variation, with certain seeds resulting in more frequent long delays.

11. Balked Customers:
- The number of balked customers ranges from 17 to 44.
- The variance (80.04) indicates considerable fluctuation in the number of customers who leave due to a full queue.

Conclusion

The simulation results show that while most average performance measures (such as delay, queue length, and utilization) are relatively stable across different random

seeds, the extreme values (like maximum delay and maximum time in system) exhibit more variability. The queue capacity limit of 2 consistently leads to balking, with the number of balked customers varying significantly depending on the random seed.

Summary

This report presents the simulation and analysis of a single-server queue (M/M/1 queue) with FIFO discipline, focusing on various operational conditions and performance measures. The objectives of the experiment include executing the basic simulation, assessing the impact of random seed variation, modifying performance measures, adjusting operational time, introducing queue capacity limits, and developing an integrated visual program for running the simulations. The following summarizes the key findings from each part of the experiment:

1. Simulation Execution:
- The basic M/M/1 queue simulation was implemented and executed. The average delay in queue, average number of customers in queue, and server utilization were calculated.
- The results demonstrated consistent performance measures with minimal variance, indicating stable and predictable queue behavior under fixed conditions.

2. Random Seed Variation:
- The simulation was run 10 times with different random seeds to assess variability in the outputs.
- While the results showed some variation due to the inherent randomness in the arrival and service processes, the average metrics remained within a similar range, suggesting robust system performance.

3. Performance Measures Modification:
- The simulation code was modified to compute additional performance metrics, including the time-average number of customers in the system, average total time in the system, maximum queue length, maximum delay in queue, maximum time in system, and the proportion of customers delayed over 1 minute.
- These enhanced metrics provided deeper insights into system congestion, customer experience, and worst-case scenarios.

4. Operational Time Adjustment:
- The simulation was adjusted to reflect a fixed operational time from 9 A.M. to 5 P.M., continuing to serve all customers present at closing time.
- The analysis showed that the system performed efficiently with predictable performance metrics, despite some variability in peak conditions and queue lengths.

5. Queue Capacity Limitation:
- A queue capacity limit of two customers was introduced, resulting in customers

balking when the queue was full.

- The simulation results indicated significant fluctuations in the number of balked customers, with noticeable impacts on performance measures such as maximum delay and maximum time in the system.

6. Integrated Visual Program:

- The original C++ program was modified to develop an integrated visual program using the MFC library, facilitating interactive simulations and real-time display of results.

Analysis of Results

1. Fixed Customers Condition:

- The average delay in queue and other performance metrics showed minimal variance across multiple runs with the same seed, indicating stable queue behavior.
- With different seeds, the metrics varied slightly but remained within an acceptable range, reflecting the stochastic nature of the system.

2. Limited Service Time Condition:

- The results from simulations with a fixed operational time (9 A.M. to 5 P.M.) showed relatively stable average performance metrics.
- Peak conditions and extreme values exhibited more variability, highlighting the impact of random arrivals and service times on the system.

3. Queue Capacity Limitation:

- Introducing a queue capacity limit resulted in significant variability in the number of balked customers, affecting other performance measures.
- The maximum queue length consistently reached the capacity limit, indicating frequent occurrences of full queues and customer balking.