CSCI 140 PA 6 Submission

Due Date: 10/07/2021 Late (date and time):________________

Name(s): Nero Li

Exercise 1 (**with extra credit**) -- need to submit source code and I/O

-- check if completely done ✓; otherwise, discuss issues below

Source code below:

```
/*  Program: PA_6_exercise_1
    Author: Nero Li
    Class: CSCI 220
    Date: 10/08/2021
    Description:
        You need to write a program that simulates call lines at a calling
center. A
        line is basically a queue object, and you can assume one customer
representative
        per line. Based on some preliminary estimate, customers are
expected to call
        in random integer intervals of 1 to n minutes (inclusively and two
customers
        are expected to call between that interval so if customer 1 calls
at minute 3,
        then customer 2 is expected to call between minute 4 and 7 when n
is 4). Also,
        it is expected to take a random integer interval of 1 to m minutes
to serve
        each customer. If the call arrival rate is faster than the service
rate (n<m),
        the line will grow indefinitely. Even with a "balanced" rate (n =
m), randomness
        can still possibly cause a long line over a period. You are going
to run the
        calling center simulation with one line (l = 1) for a period of t
minutes using
        the following algorithm (enter input parameters from keyboard and
use l = 1,
        t = 30, n = 4, and m = 4 as test case 0)

    I certify that the code below is my own work.

      Exception(s): N/A

*/

#include <iostream>
```

```
#include <fstream>
#include <ctime>
#include <queue>

using namespace std;

int randRange(const unsigned int& start, const unsigned int& end)
{
    int scalingFactor = end - start + 1;
    return rand() % scalingFactor + start;
}

void func(string str, int l, int t, int n, int m)
{
    ofstream fout;
    queue<int> customerLine[4];
    queue<int> startWaitingTime[4];
    bool serving{false};
    int customerID{1};
    int totalServed{0};
    int callTime{randRange(1, n)};
    int serveEnd{0};
    size_t maxLineSize{0};
    float curWaitTime{0};
    float maxWaitTime{0};
    float totWaitTime{0};
    float avgWaitTime{0};

    fout.open(str, ios::binary);
    --l;

    cout << str << endl;
    for (int i = 1; i <= t; ++i)
    {
        if (i <= 15 || i >= (t - 15))
        {
            cout << "Minute: " << i << endl;
            fout << "Minute: " << i << endl;
        }
        if (i == callTime)
        {
            if (i <= 15 || i >= (t - 15))
            {
                cout << "Customer #" << customerID << " calls..." << endl;
                fout << "Customer #" << customerID << " calls..." << endl;
            }
            customerLine[l].push(customerID);
            startWaitingTime[l].push(i);
            callTime += randRange(1, n);
            if (customerLine[l].size() > maxLineSize)
            {
                maxLineSize = customerLine[l].size();
            }

            ++customerID;
        }
        if (i == serveEnd)
        {
            if (i <= 15 || i >= (t - 15))
            {
                cout << "Service was just completed for a customer..." <<
endl;
                fout << "Service was just completed for a customer..." <<
endl;
            }
            serving = false;
            ++totalServed;
        }
        if (!serving && !customerLine[l].empty())
        {
            if (i <= 15 || i >= (t - 15))
            {
                cout << "Start serving customer #" <<
customerLine[l].front() << "..." << endl;
                fout << "Start serving customer #" <<
customerLine[l].front() << "..." << endl;
            }

            curWaitTime = i - startWaitingTime[l].front();
            if (curWaitTime > maxWaitTime)
            {
                maxWaitTime = curWaitTime;
            }
            totWaitTime += curWaitTime;

            customerLine[l].pop();
            startWaitingTime[l].pop();
            serving = true;
            serveEnd = i + randRange(1, m);
        }

        if (customerLine[l].size() > maxLineSize)
        {
            maxLineSize = customerLine[l].size();
        }

    }

    avgWaitTime = totWaitTime / totalServed;

    cout << "--------------------Result--------------------" << endl;
    cout << "Number of customers served: " << totalServed << endl;
    cout << "Number of customers left in queue: " <<
customerLine[l].size() << endl;
```

```
    cout << "Maximum number of customers in the line at any time: " <<
maxLineSize << endl;
    cout << "Longest wait time that a customer experiences before being
served: " << maxWaitTime << endl;
    cout << "Average wait time for all customers that were served: " <<
avgWaitTime << endl;

    fout << "--------------------Result--------------------" << endl;
    fout << "Number of customers served: " << totalServed << endl;
    fout << "Number of customers left in queue: " <<
customerLine[1].size() << endl;
    fout << "Maximum number of customers in the line at any time: " <<
maxLineSize << endl;
    fout << "Longest wait time that a customer experiences before being
served: " << maxWaitTime << endl;
    fout << "Average wait time for all customers that were served: " <<
avgWaitTime << endl;
    fout << endl << "Author: Nero Li\n";
}

int main()
{
    srand(time(NULL));

    func("test_case_0.txt", 1, 30, 4, 4);
    func("test_case_1.txt", 1, 360, 4, 4);
    func("test_case_2.txt", 1, 360, 3, 4);
    func("test_case_3.txt", 1, 360, 5, 4);

    cout << "Author: Nero Li\n";
    return 0;
}
```

Input/output below:

```
test_case_0.txt
Minute: 1
Minute: 2
Customer #1 calls...
Start serving customer #1...
Minute: 3
Minute: 4
Service was just completed for a customer...
Minute: 5
Minute: 6
Customer #2 calls...
Start serving customer #2...
Minute: 7
Service was just completed for a customer...
Minute: 8
Minute: 9
Minute: 10
Customer #3 calls...
```

```
Start serving customer #3...
Minute: 11
Minute: 12
Minute: 13
Service was just completed for a customer...
Minute: 14
Customer #4 calls...
Start serving customer #4...
Minute: 15
Service was just completed for a customer...
Minute: 16
Minute: 17
Minute: 18
Customer #5 calls...
Start serving customer #5...
Minute: 19
Service was just completed for a customer...
Minute: 20
Minute: 21
Customer #6 calls...
Start serving customer #6...
Minute: 22
Minute: 23
Minute: 24
Minute: 25
Customer #7 calls...
Service was just completed for a customer...
Start serving customer #7...
Minute: 26
Service was just completed for a customer...
Minute: 27
Customer #8 calls...
Start serving customer #8...
Minute: 28
Minute: 29
Customer #9 calls...
Service was just completed for a customer...
Start serving customer #9...
Minute: 30
--------------------Result--------------------
Number of customers served: 8
Number of customers left in queue: 0
Maximum number of customers in the line at any time: 1
Longest wait time that a customer experiences before being served: 0
Average wait time for all customers that were served: 0
test_case_1.txt
Minute: 1
Minute: 2
Minute: 3
Minute: 4
Customer #1 calls...
Start serving customer #1...
```

```
Minute: 5
Minute: 6
Service was just completed for a customer...
Minute: 7
Minute: 8
Customer #2 calls...
Start serving customer #2...
Minute: 9
Minute: 10
Minute: 11
Service was just completed for a customer...
Minute: 12
Customer #3 calls...
Start serving customer #3...
Minute: 13
Service was just completed for a customer...
Minute: 14
Minute: 15
Customer #4 calls...
Start serving customer #4...
Minute: 345
Service was just completed for a customer...
Start serving customer #128...
Minute: 346
Customer #136 calls...
Minute: 347
Service was just completed for a customer...
Start serving customer #129...
Minute: 348
Minute: 349
Customer #137 calls...
Minute: 350
Minute: 351
Service was just completed for a customer...
Start serving customer #130...
Minute: 352
Customer #138 calls...
Minute: 353
Minute: 354
Minute: 355
Customer #139 calls...
Service was just completed for a customer...
Start serving customer #131...
Minute: 356
Minute: 357
Minute: 358
Customer #140 calls...
Minute: 359
Service was just completed for a customer...
Start serving customer #132...
Minute: 360
--------------------Result--------------------
```

```
Number of customers served: 131
Number of customers left in queue: 8
Maximum number of customers in the line at any time: 12
Longest wait time that a customer experiences before being served: 26
Average wait time for all customers that were served: 13.916
test_case_2.txt
Minute: 1
Customer #1 calls...
Start serving customer #1...
Minute: 2
Customer #2 calls...
Minute: 3
Service was just completed for a customer...
Start serving customer #2...
Minute: 4
Minute: 5
Customer #3 calls...
Minute: 6
Minute: 7
Customer #4 calls...
Service was just completed for a customer...
Start serving customer #3...
Minute: 8
Customer #5 calls...
Minute: 9
Customer #6 calls...
Minute: 10
Service was just completed for a customer...
Start serving customer #4...
Minute: 11
Customer #7 calls...
Minute: 12
Customer #8 calls...
Minute: 13
Minute: 14
Service was just completed for a customer...
Start serving customer #5...
Minute: 15
Customer #9 calls...
Service was just completed for a customer...
Start serving customer #6...
Minute: 345
Customer #172 calls...
Service was just completed for a customer...
Start serving customer #137...
Minute: 346
Minute: 347
Minute: 348
Customer #173 calls...
Service was just completed for a customer...
Start serving customer #138...
Minute: 349
```

```
Minute: 350
Customer #174 calls...
Service was just completed for a customer...
Start serving customer #139...
Minute: 351
Customer #175 calls...
Minute: 352
Service was just completed for a customer...
Start serving customer #140...
Minute: 353
Minute: 354
Customer #176 calls...
Service was just completed for a customer...
Start serving customer #141...
Minute: 355
Minute: 356
Minute: 357
Customer #177 calls...
Service was just completed for a customer...
Start serving customer #142...
Minute: 358
Customer #178 calls...
Service was just completed for a customer...
Start serving customer #143...
Minute: 359
Minute: 360
Service was just completed for a customer...
Start serving customer #144...
--------------------Result--------------------
Number of customers served: 143
Number of customers left in queue: 34
Maximum number of customers in the line at any time: 37
Longest wait time that a customer experiences before being served: 70
Average wait time for all customers that were served: 37.5175
test_case_3.txt
Minute: 1
Minute: 2
Customer #1 calls...
Start serving customer #1...
Minute: 3
Service was just completed for a customer...
Minute: 4
Minute: 5
Customer #2 calls...
Start serving customer #2...
Minute: 6
Customer #3 calls...
Minute: 7
Service was just completed for a customer...
Start serving customer #3...
Minute: 8
Service was just completed for a customer...
```

```
Minute: 9
Minute: 10
Customer #4 calls...
Start serving customer #4...
Minute: 11
Customer #5 calls...
Service was just completed for a customer...
Start serving customer #5...
Minute: 12
Service was just completed for a customer...
Minute: 13
Customer #6 calls...
Start serving customer #6...
Minute: 14
Service was just completed for a customer...
Minute: 15
Minute: 345
Minute: 346
Service was just completed for a customer...
Start serving customer #118...
Minute: 347
Minute: 348
Service was just completed for a customer...
Minute: 349
Customer #119 calls...
Start serving customer #119...
Minute: 350
Customer #120 calls...
Minute: 351
Service was just completed for a customer...
Start serving customer #120...
Minute: 352
Minute: 353
Minute: 354
Customer #121 calls...
Minute: 355
Customer #122 calls...
Service was just completed for a customer...
Start serving customer #121...
Minute: 356
Service was just completed for a customer...
Start serving customer #122...
Minute: 357
Customer #123 calls...
Service was just completed for a customer...
Start serving customer #123...
Minute: 358
Minute: 359
Service was just completed for a customer...
Minute: 360
--------------------Result--------------------
Number of customers served: 123
Number of customers left in queue: 0
Maximum number of customers in the line at any time: 6
Longest wait time that a customer experiences before being served: 9
Average wait time for all customers that were served: 1.60163
Author: Nero Li
```

Exercise 2 -- need to submit source code and I/O
-- check if completely done ✓; otherwise, discuss issues below
Source code below:

```
/*  Program: PA_6_exercise_2
    Author: Nero Li
    Class: CSCI 220
    Date: 10/08/2021
    Description:
        Provide a linked implementation of a deque and name it LinkedDeque
(use
        doubly linked list). It can be a template/generic class, or you
can set
        it up with a certain data type like string. Use a test_1 driver to
try out
        your LinkedDeque by adding and removing values from both ends. Try
the
        following test_1 cases: insert front, insert front, insert rear,
remove rear,
        remove rear, size, and front item.

    I certify that the code below is my own work.

      Exception(s): N/A

*/

#include <iostream>
#include <exception>

using namespace std;

class RuntimeException { // generic run-time exception
private:
      string errorMsg;
public:
      RuntimeException(const string& err) { errorMsg = err; }
      string getMessage() const { return errorMsg; }
};

class DequeEmpty : public RuntimeException {
public:
    DequeEmpty(const string& err) : RuntimeException(err) { }
};

template <typename T>
```

```
class LinkedDeque
{
public:
    LinkedDeque()
    : head(NULL), tail(NULL), amount(0) {}

    bool empty()
    {
        return ((amount == 0) ? true : false);
    }

    int size()
    {
        return amount;
    }

    T front()
    {
        errCheck();
        return head->n;
    }

    T rear()
    {
        errCheck();
        return tail->n;
    }

    void insertFront(T n)
    {
        Node *newNode = new Node;
        newNode->n = n;
        newNode->prev = NULL;
        if (empty())
        {
            newNode->next = NULL;
            head = newNode;
            tail = newNode;
        }
        else
        {
            newNode->next = head;
            head->prev = newNode;
            head = newNode;
        }
        ++amount;
    }

    void insetRear(T n)
    {
        Node *newNode = new Node;
        newNode->n = n;
        newNode->next = NULL;
        if (empty())
        {
            newNode->prev = NULL;
            head = newNode;
            tail = newNode;
        }
        else
        {
            newNode->prev = tail;
            tail->next = newNode;
            tail = newNode;
        }
        ++amount;
    }

    T removeFront()
    {
        errCheck();
        T n;
        Node *del = head;
        n = head->n;
        head = head->next;
        delete(del);
        --amount;
        return n;
    }

    T removeRear()
    {
        errCheck();
        T n;
        Node *del = tail;
        n = tail->n;
        tail = tail->prev;
        delete(del);
        --amount;
        return n;
    }

private:
    struct Node
    {
        T n;
        Node *prev;
        Node *next;
    } *head, *tail;
    int amount;

    void errCheck()
    {
        if (amount == 0)
```

```
            throw DequeEmpty("No elements in the queue.");
    }
};

int main()
{
    LinkedDeque<string> test_1;
    LinkedDeque<string> test_2;

    test_1.insertFront("Second");
    test_1.insertFront("First");
    test_1.insetRear("Third");
    cout << test_1.removeRear() << endl;
    cout << test_1.removeRear() << endl;
    cout << test_1.size() << endl;
    cout << test_1.front() << endl;

    test_2.insetRear("Fourth");
    test_2.insetRear("Fifth");
    test_2.insertFront("Third");
    test_2.insertFront("Second");
    test_2.insertFront("First");
    cout << test_2.size() << endl;
    cout << test_2.removeFront() << endl;
    cout << test_2.removeFront() << endl;
    cout << test_2.removeFront() << endl;
    cout << test_2.removeFront() << endl;
    cout << test_2.rear() << endl;

    cout << "Author: Nero Li\n";
    return 0;
}
```

Input/output below:

```
Third
Second
1
First
5
First
Second
Third
Fourth
Fifth
Author: Nero Li
```

Answer for Question 1:

I wouldn't do an array implementation for deque. Since we need to change two sides of the queue, a doubled linked list can easily do push function or pop function by changing the head and the tail for that linked list. For array, each time you pop and push a new variable at the front, you need to modify your whole array in order to get or remove the space for that variable.

Answer for Question 2:

If we want to use deque as a queue, when we insert something at front, the function insertFront() will become enqueue(), and we can only do removeRear() to remove element at rear as dequeue(); when we insert something at rear, the function insertRear() will become enqueue(), and we can only do removeFront() to remove element at front as dequeue().