ECE 36800 – Data Structures
H-Option Programming Assignment on DES

This Assignment:
You will write a program to simulate a queuing system and estimate the system performance using a discrete event simulator.

Purpose:
Ensure that you can create and manipulate the queue and list container classes in the C++ Standard Template Library.
Understand and practice discrete-event simulation (DES) in a real application.

Files that you must write:
1. `customer.h`: header file for `customer` class. Start with the given version and add public functions that you think will be useful. You may create a `customer.cpp` file if needed.
2. `washevent.h`: The header file for the `washevent` class. Start with the given version and add public functions that you think will be useful. You may create a `washevent.cpp` file if needed
3. `carwash.cpp`: Your test program performing the simulation of the car wash queuing system. Start with the given file and add your implementation.

Guideline:
In this project, you will write a program to simulate the car wash service at a car wash center. Assume that there is only one car wash line for customers; all customers must wait in one line for service. Use the `queue` container class in the C++ Standard Template Library for the service waiting line, so that the service center can serve customer in a *First Come First Serve (FCFS)* policy. Calculate the average number of customers in the waiting line (including the customer currently under service) and the average time that a customer spends waiting for service to finish (including the time spent in queue and the service time).

In order to simulate the service, your program will first read the following parameters from the standard input:

- *ave_interarrival_time*: the average time interval between two consecutive arrivals. The actual interval between two consecutive arrivals should be a random number in (0, 2**ta*).
- *ave_service_time*: the average service time. The actual service time of a customer should be a random number between (0, 2**ts*).
- *total_simulation_time*: the total simulation time should be fairly large to get reasonable estimations of the average values. A suggested simulation time should be at least 10000**ave_interarrival_time*.

The user has two options to display some debugging messages. One option determines whether your program prints the customers in the waiting line each time the queue is changed. The other option determines whether your program prints every customer's arrival and departure. After the simulation time, your program should print the average number of customers in the queue and the average waiting time that a customer spends in the queue.

Below is an example of expected output of your program. Here we choose a short simulation time to show all debugging messages.

```
Input the average time interval of two consecutive arrivals: 10
Input the average service time: 6
Input the simulation time: 100
Debugging option:
- Print items in queue at every event? (y/n): y
- Print events? (y/n): y

[debug] C1 arrives at time 2.11
[debug] Queue at time 2.11 (from rear to front): C1
[debug] C1 leaves at time 8.02
[debug] Queue at time 8.02 (from rear to front): empty
[debug] C2 arrives at time 21.35
[debug] Queue at time 21.35 (from rear to front): C2
[debug] C3 arrives at time 24.14
[debug] Queue at time 24.14 (from rear to front): C3 C2
[debug] C2 leaves at time 25.63
[debug] Queue at time 25.63 (from rear to front): C3
[debug] C4 arrives at time 26.27
[debug] Queue at time 26.27 (from rear to front): C4 C3
[debug] C3 leaves at time 35.41
[debug] Queue at time 35.41 (from rear to front): C4
[debug] C5 arrives at time 38.09
[debug] Queue at time 38.09 (from rear to front): C5 C4
[debug] C4 leaves at time 44.23
[debug] Queue at time 44.23 (from rear to front): C5
[debug] C5 leaves at time 49.04
[debug] Queue at time 49.04 from rear to front): empty
[debug] C6 arrives at time 58.36
[debug] Queue at time 58.36 (from rear to front): C6
[debug] C6 leaves at time 66.55
[debug] Queue at time 66.55 (from rear to front): empty
[debug] C7 arrives at time 70.46
[debug] Queue at time 70.46 (from rear to front): C7
[debug] C8 arrives at time 74.89
[debug] Queue at time 74.89 (from rear to front): C8 C7
[debug] C7 leaves at time 78.38
[debug] Queue at time 78.38 (from rear to front): C8
[debug] C9 arrives at time 90.25
[debug] Queue at time 90.25 (from rear to front): C9 C8
[debug] C8 leaves at time 90.98
[debug] Queue at time 90.98 (from rear to front): C9
[debug] C9 leaves at time 94.57
[debug] Queue at time 94.57 (from rear to front): empty
[debug] end of simulation at time 100

The average number of customers in the queue is 0.8689
The average waiting time for a customer is 9.6544
```

Discussions:

1. Use the `list` container class in in the C++ Standard Template Library to maintain a list of `washevent` that record the three different types of `washevents`: *arrival*, *departure*, and *end_of_simulation*. The `list` can be sorted in ascending order of time using its member function `sort`, with a Boolean function `comp` as input to compare elements. The `comp` function is provided in `washevent.h`.
2. Each time a new `washevent` is retrieved from the event list, your program will take different actions correspondingly. For example, update the queue, record statistics, and schedule new `washevents`.
3. In order to calculate the average waiting time, you need to record the number of customers whose service have been completed and record the total waiting time.
4. Please use the following method to calculate the average number of customers in the queue: Record the time whenever a change in the queue size occurs. Suppose that a queue length q_n has lasted for a time period of length t_n, and the total simulation time is $T = \sum_n t_n$, then the average queue length $=\sum_n \frac{q_n t_n}{T}$.
5. Create two random number generators: one generates the arrival time intervals; the other generates random values for the service time; both need to be of `double` type. The following example shows how to generate a uniformly distributed random variable *a* with mean *r*.

```
#include <random>
// include the pseudo-random number generation library

int main()
{
    random_device rd;
    default_random_engine gen(rd());
    double r=10.0;
    uniform_real_distribution<double> dis(0.0, 2*r);
    // Define a uniform_real_distribution class dis
    // to be used later to generate random values.

    ....

    double a = dis(gen);
    // return a double-precision floating-point value
    // that uniformly distributed between [0, 2*r].

    return 0;
}
```

The above example requires your C++ compiler follows C++11 or later version. If you use gcc to compile, make sure you add the `-std=c++11` option, e.g.,

```
g++ -g -Wall -std=c++11 carwash.cc customer.h
washevent.h -o carwash
```

Grading Policy:

There are 100 points for this project.

Please make sure your program compiles successfully. If not, 50 points will be deducted.

1. **Executability** (10%):
   - Runtime errors: You program must not have runtime errors (e.g., code crash, infinite loop, reading uninitialized memory, accessing the content of a NULL pointer, etc.).
2. **Programming style** (5%):
   - Code efficiency: Code should use the best approach in every case.
   - Readability: Code should be clean, understandable, and well-organized. Please pay special attention to indentation, use of whitespace, variable naming, and organization.
   - Documentation: Code should be well-commented with file header and comments.
3. **Program Specifications/Correctness** (80%):

   Please refer to the Grading Criteria table for details. Specifically, your program should behave correctly, adhere to the instructions, and pass the test program.

	points	
first arrival -> event list	5	
end_of_simulation -> event list	5	
arrival event handling	20	
create a new customer -> queue		5
if queue is empty		
update departure time		5
departure -> event list		5
new arrival -> event list		5
departure event handling	15	
pop the front customer		5
if queue is not empty		
update departure time of next customer		5
departure -> event list		5
end_of_simulation handling	5	
print statistics		5
debug options	10	
print queue		5
print arrival and departure		5
statistics	20	
average queue length		10
average waiting time		10
total	80	