#==============================#

| Yuan Wang, Yizhen Wang |

| ywang19, ywang22 |

| PA4 README |

#==============================#

#===================#

| Summary |

#===================#

**This is the event driven simulation problem which is about PA4.**

1. The Standard Command Line input is : ./qSim [#customers] [#tellers] [#simulationTime] [#averageServiceTime]
2. Note: After some tests, I found that it will give more consistent results in **seconds (int)** than computing with floating numbers.
3. My classes are named differently than the assignment specifies:
4. Customer -> CustomerEvent;
5. Teller -> TellerEvent;
6. Event -> BankEvent
7. EventQueue -> queue (not a struct)
8. The “Action” method is called onCompletion.(B/C it is more directly)

The whole program includes:

1. Event.h:

Defined the EventQueue(linked-list), BankEvent, TellerEvent,CustomerEvent.

1. Event.cpp

Defined the setters and getters of the TellerEvent and CustomerEvent, and the Queue operation functions.

1. function.cpp

Defined some the help functions

1. test.cpp

It is the main function.

Defined :

runMultiQueueSimulation()>>multi Queue

runUniQueueSimulation() >> Single

TellerEvent::onCompletion(EventQueue\* e)//action method for Tellerevent

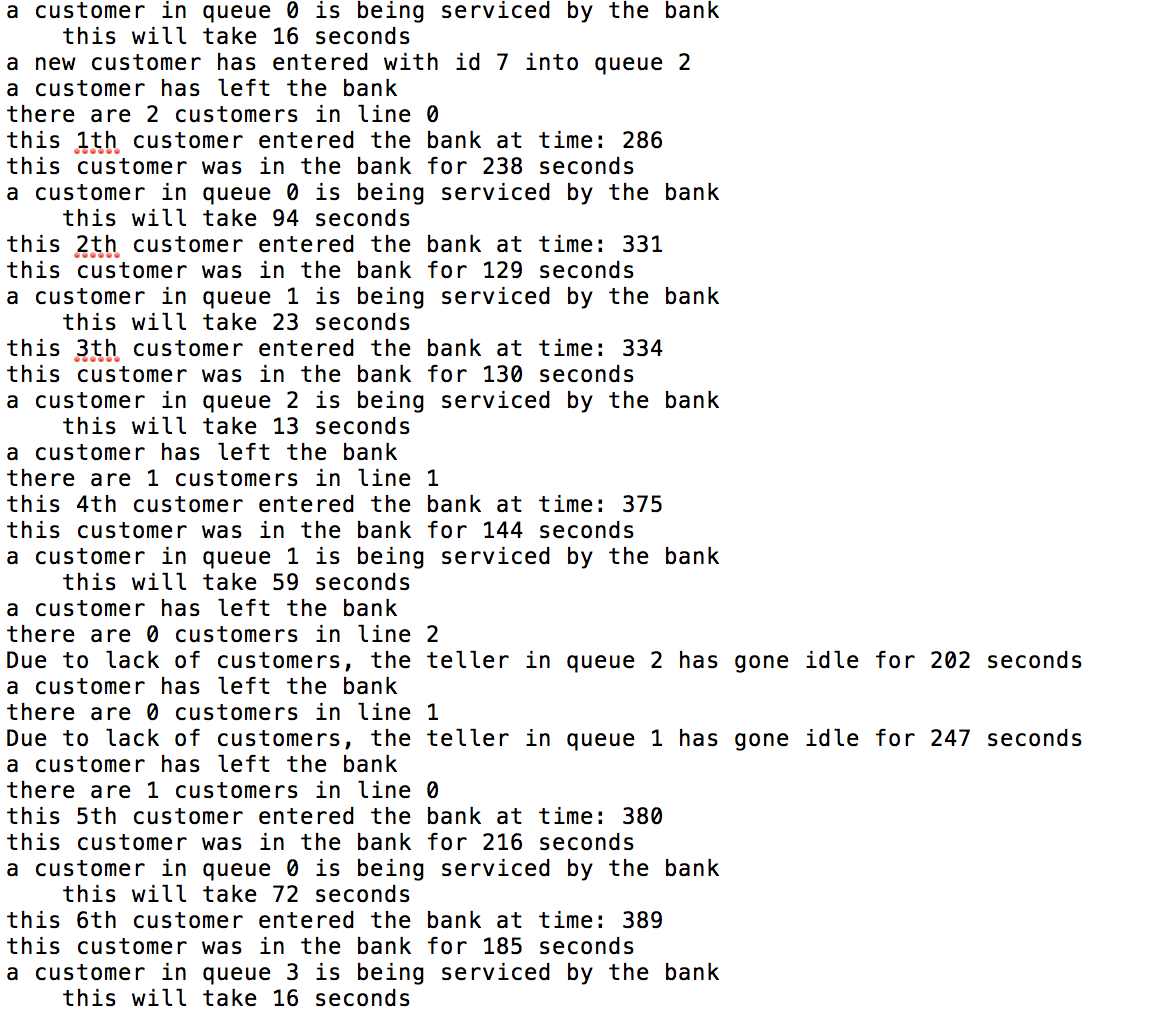
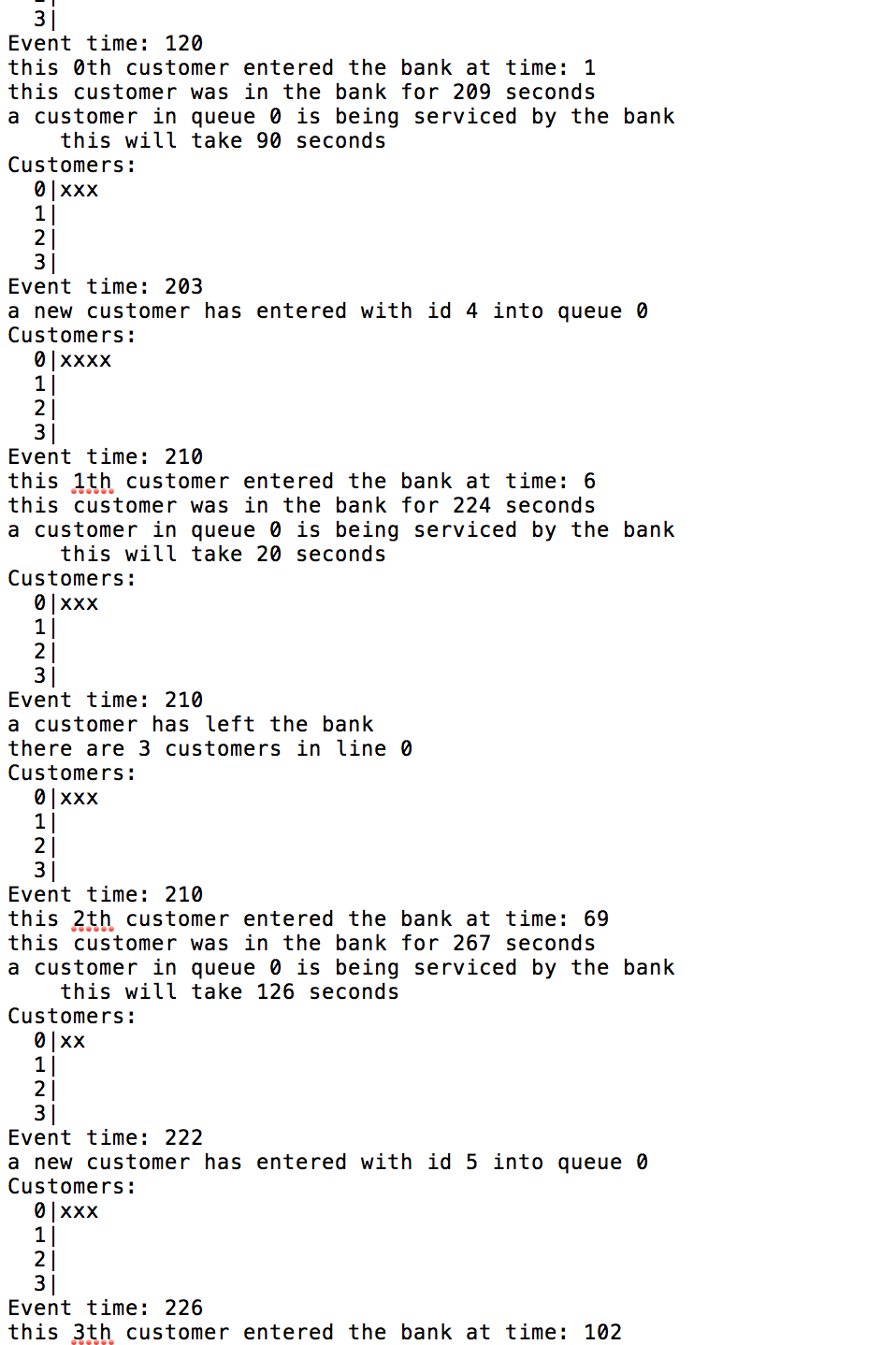
CustomerEvent::onCompletion(EventQueue\* e)//action method for CustomerEvent

BankEvent::onCompletion(EventQueue\* e)//action method for BankEvent

Other help functions….

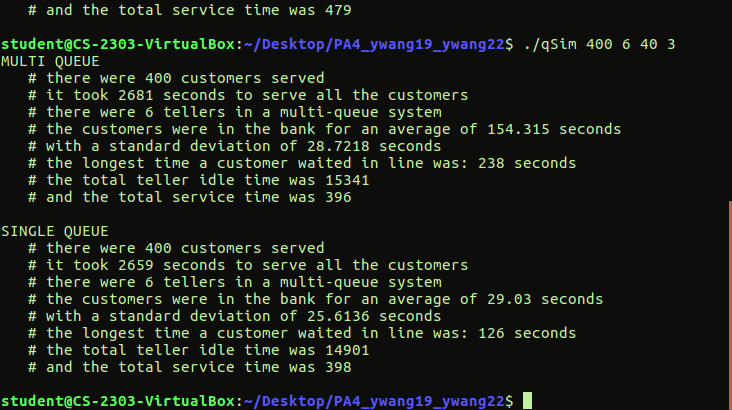
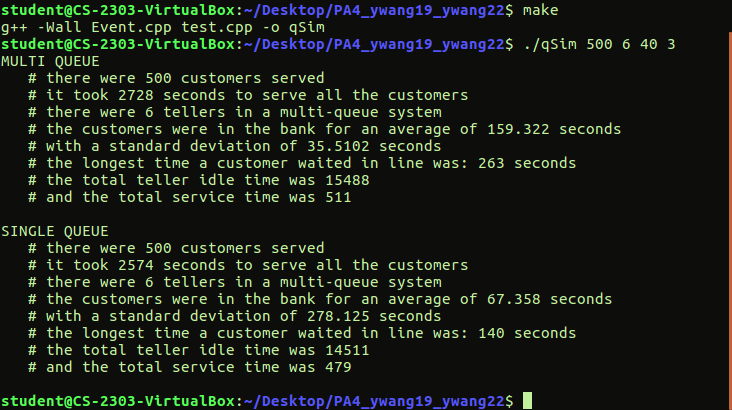
About the onCompletition part, I have defined a DEBUG to show the actual process model about the event. It is just for my own interests.

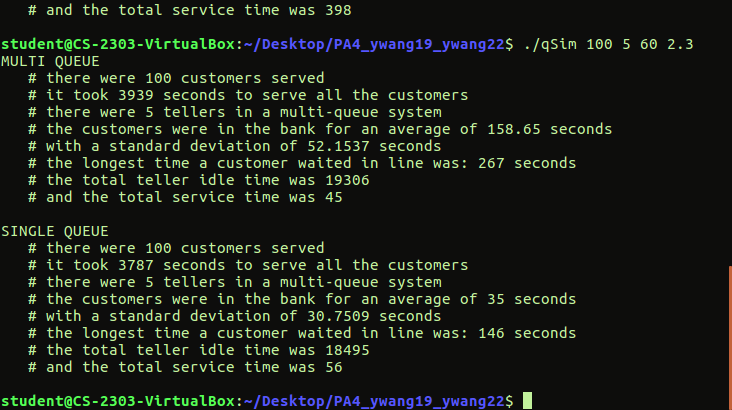
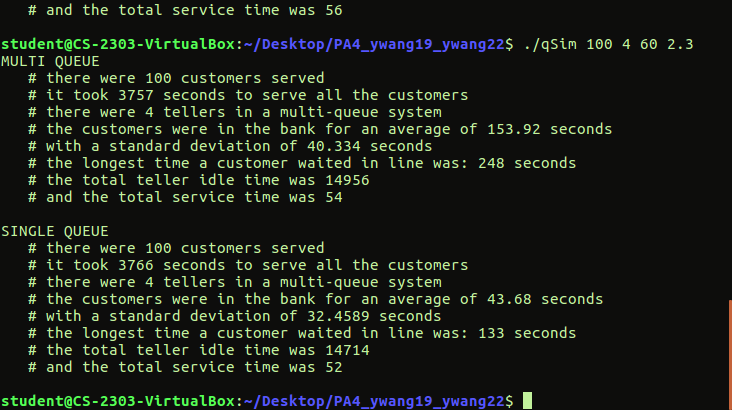
Here is the screenshot about the debug process.



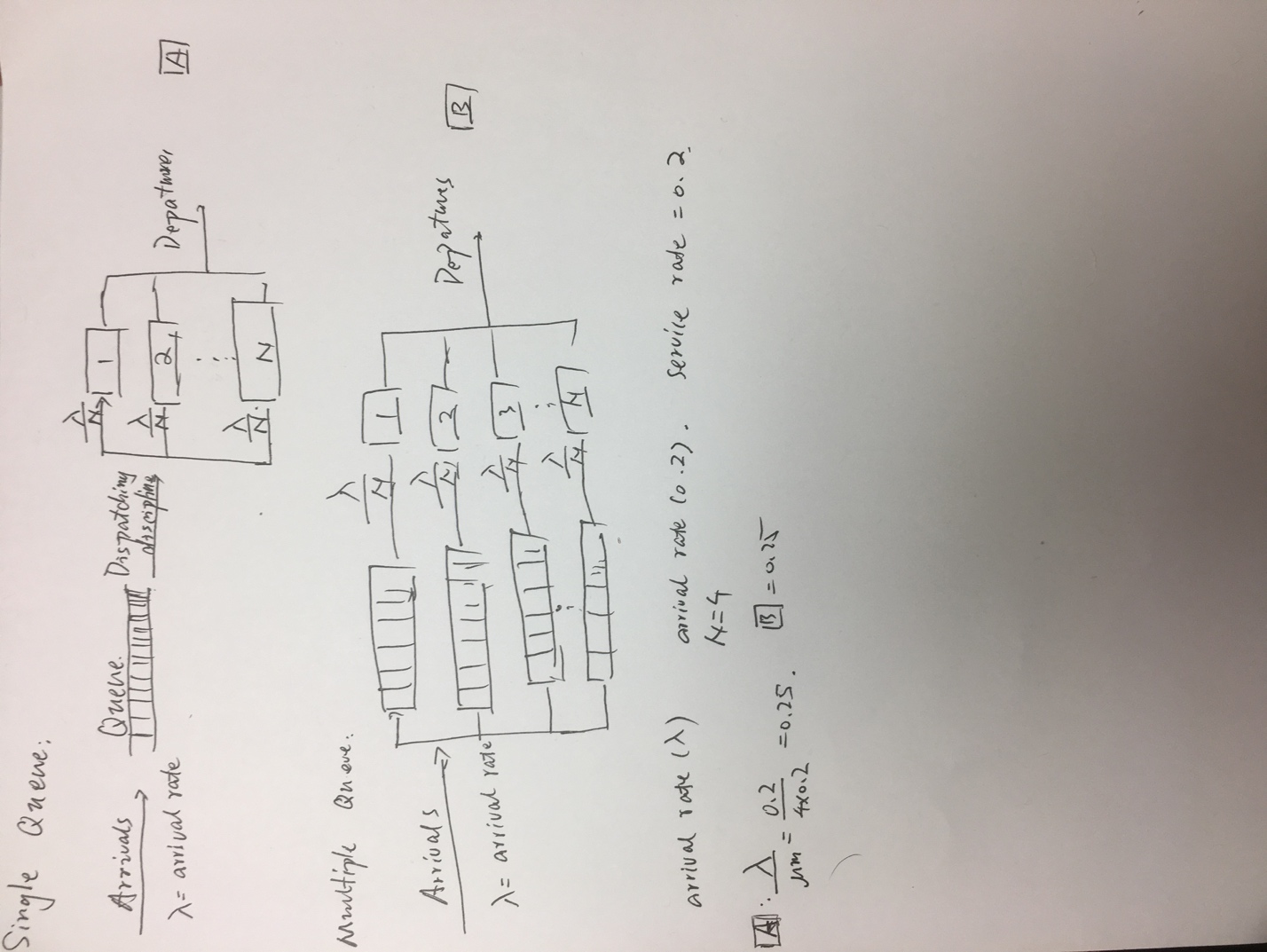
Outputs:

Here are the four outputs from the command line.





Analyze:



Here is the model of the single and multiple queue, and combine with the output results. Single Queue has a little advantage than the multiple. Generally, they are almost the same.