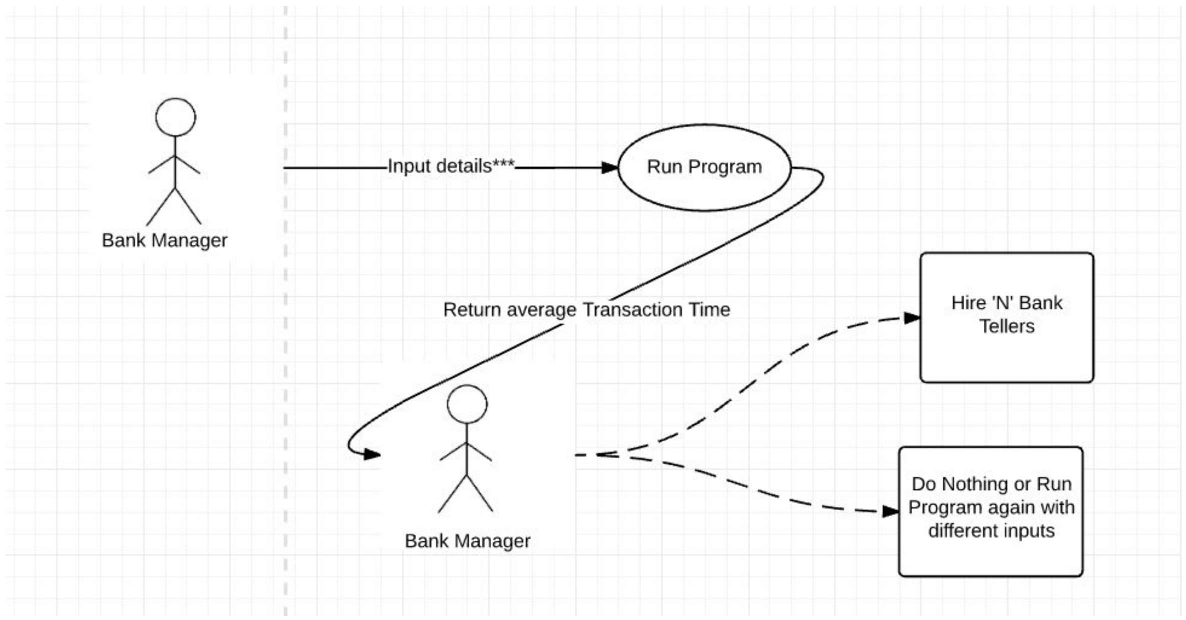
Brian Faure
RUID:150003563
NetID:bdf 39
Programming Methodology II
Project #1, Fall 2015

(I.) Formulate the problem precisely and define the use case. [1 point]

From my interpretation of the given problem, we must write a program for a bank manager which will allow him/her to test out whether or not it will be efficient to hire more bank tellers. The bank manager is evaluating the efficiency of the hiring process based on the increase in customer satisfaction as a product of the new bank tellers vs. the cost of hiring the additional workers. The increase in customer satisfaction, in this case, can be attributed directly to the amount of the time customers must wait in line when trying to fulfill a transaction at the bank itself.

As a summary of the Use Case, the bank manager will model the arrival of bank customers by, entering into the program, the arrival times and transaction times for a certain number of customers for a set amount of time. The bank teller will then run the program by inputting the number of bank tellers he/she wishes to have working total and hitting ENTER. At this point the program will model the entrance and departure of all of the customers entered by the bank manager while also ensuring to record the total time they needed to stay at the bank to finished their transaction. Once the program has finished the simulation and entered the total transaction time for every customer, it will calculate the average transaction time for the entire set of people and return that output to the bank manager. At this point, the Bank Manager uses their knowledge of the workings of the bank as well as the program output to decide whether or not to hire the number or extra bank tellers he/she input into the initial program.

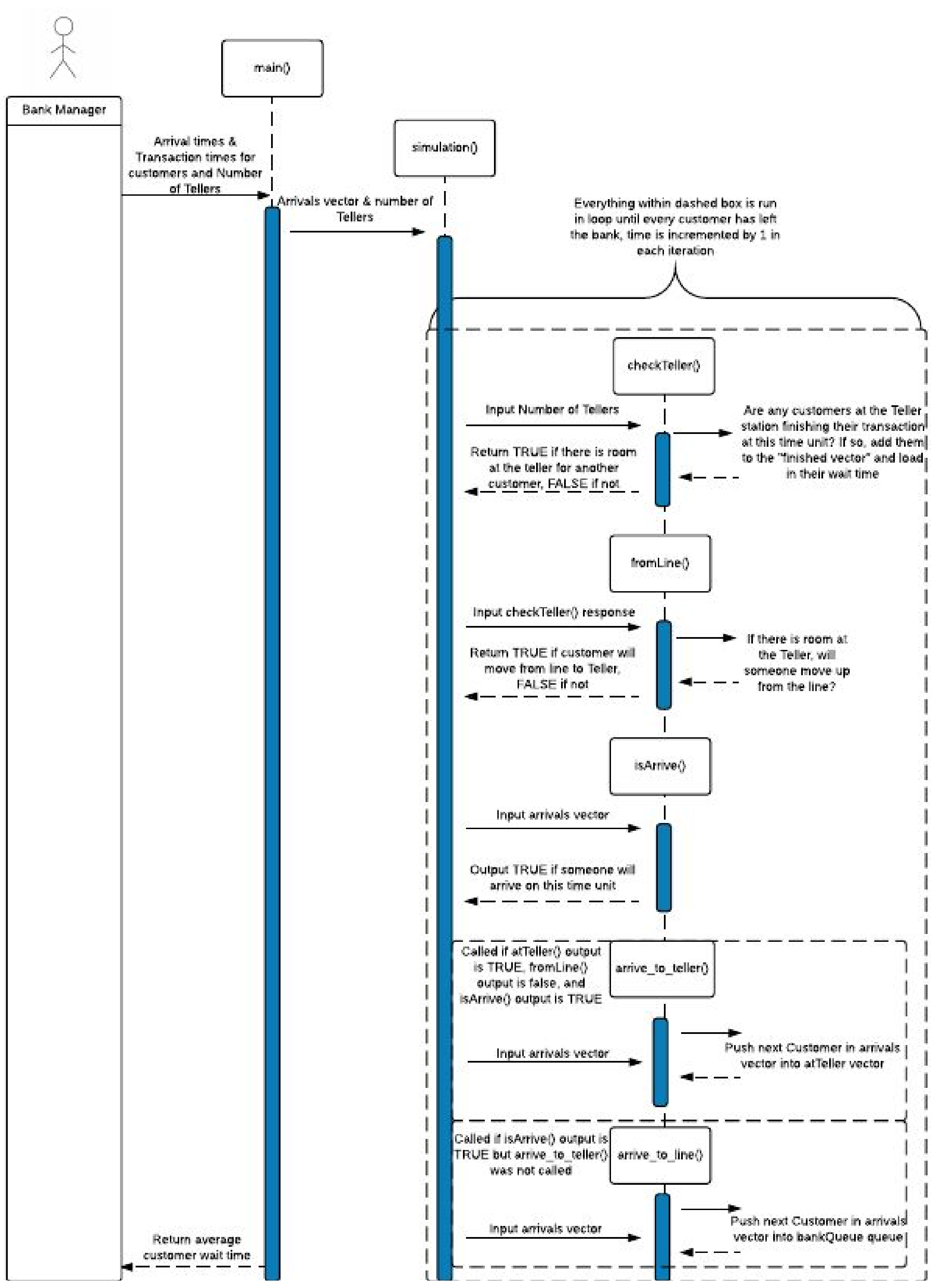

Bank Manager
Input details***
Run Program
Return average Transaction Time
Hire 'N' Bank Tellers
Bank Manager
Do Nothing or Run Program again with different inputs

***Inputs are → Arrival times & transaction times for customers, # of bank tellers (N)

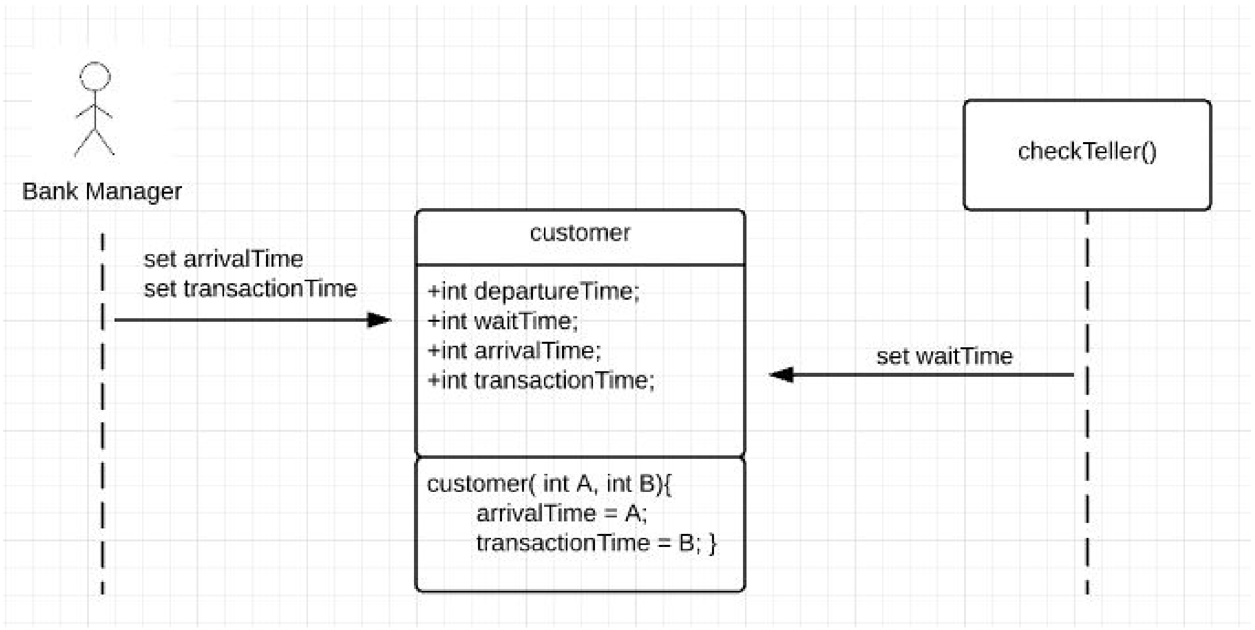
(II.) Use UML to identify user scenarios, interaction/sequence diagrams, class diagrams etc to analyze and design a solution to the problem. [3 points]

The user scenarios are outlined in the use case diagram above, the final output of the entire sequence is either more tellers are hired or the bank manager does not hire more tellers. Although the output of the program is simply the average wait time per customer, as stated above, we assume that the bank manager uses this information to aid in their decision on whether or not to hire a certain number of additional bank teller.

The sequence diagram I have included on the next page goes into detail on exactly how the implementation will function. The action of the Bank Manager is exactly the same as in the use case but the sequence diagram adds the hidden layer behind what is going on in the software. The large dashed rectangle to the right of the *simulation()* function is meant to represent that all of its contents are called many times within a loop. This loop will increment the currentTime by 1 on each iteration and only finish once every customer has both arrived, finished their transaction, and left the bank.


Bank Manager
main()
simulation()
Arrival times & Transaction times for customers and Number of Tellers
Arrivals vector & number of Tellers
Everything within dashed box is run in loop until every customer has left the bank, time is incremented by 1 in each iteration
checkTeller()
Input Number of Tellers
Are any customers at the Teller station finishing their transaction at this time unit? If so, add them to the "finished vector" and load in their wait time
Return TRUE if there is room at the teller for another customer, FALSE if not
fromLine()
Input checkTeller() response
If there is room at the Teller, will someone move up from the line?
Return TRUE if customer will move from line to Teller, FALSE if not
isArrive()
Input arrivals vector
Output TRUE if someone will arrive on this time unit
Called if atTeller() output is TRUE, fromLine() output is false, and isArrive() output is TRUE
arrive_to_teller()
Input arrivals vector
Push next Customer in arrivals vector into atTeller vector
Called if isArrive() output is TRUE but arrive_to_teller() was not called
arrive_to_line()
Input arrivals vector
Push next Customer in arrivals vector into bankQueue queue
Return average customer wait time

I only plan on using 1 class called “customer” to hold the information pertaining to each customer. The following is a diagram of my class “customer”



The Bank Manager is the entity that initially sets all of the arrivalTime and transactionTime variables (possibly indirectly through the main function depending on how the customer info is loaded from the Bank Manager). The waitTime variable is set, once the specific customer has finished their transaction, by the checkTeller() function. The departureTime variable can be set by either the fromLine() function or the arrive_to_teller() function because those two functions are the only ones that have the power to move a customer to the Bank Teller kiosk. When a customer is moved to the Bank Teller, their departureTime can be predicted accurately by adding the currentTime to their transactionTime.

(iii.) Implement a simulation that will mimic the arrival of customers, service by the Teller and successful departure of a customer. Your simulation MUST be able to handle more than 1 teller, different service and arrival times of customers. [5 points]

→ See the Problem3.cpp file

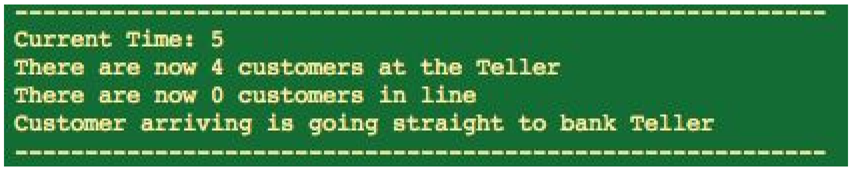
When Problem3.cpp is called, the user is prompted to enter the total # of customers they wish to enter into the simulation along with the total number of tellers. This implementation is assuming that each customer arrives in a continuous fashion (customer #1 arrives at time=1, customer #2 arrives at time=2, etc.). The transaction time for each customer is set to 10 time units.

Throughout the running of the simulation, the user is updated through the terminal on every time unit on exactly what is happening in the bank. For example, if on time unit #3 a customer is arriving and there is an opening at the teller, the simulation will print that a customer was able to arrive and move directly to the teller.

At the end of the simulation, the program outputs:

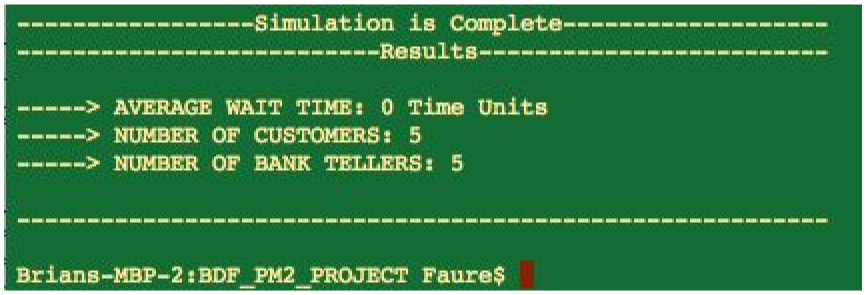
#1.) Average Wait Time (in time units)
#2.) Number of Customers
#3.) Number of Bank Tellers

→ Mid-Simulation Terminal Output Example:



```
---------------------------------------------------------
Current Time: 5
There are now 4 customers at the Teller
There are now 0 customers in line
Customer arriving is going straight to bank Teller
---------------------------------------------------------
```

→ Final Terminal Output Example:



```
-----------------Simulation is Complete-------------------
-------------------------Results--------------------------

-----> AVERAGE WAIT TIME: 0 Time Units
-----> NUMBER OF CUSTOMERS: 5
-----> NUMBER OF BANK TELLERS: 5

----------------------------------------------------------

Brians-MBP-2:BDF_PM2_PROJECT Faure$
```

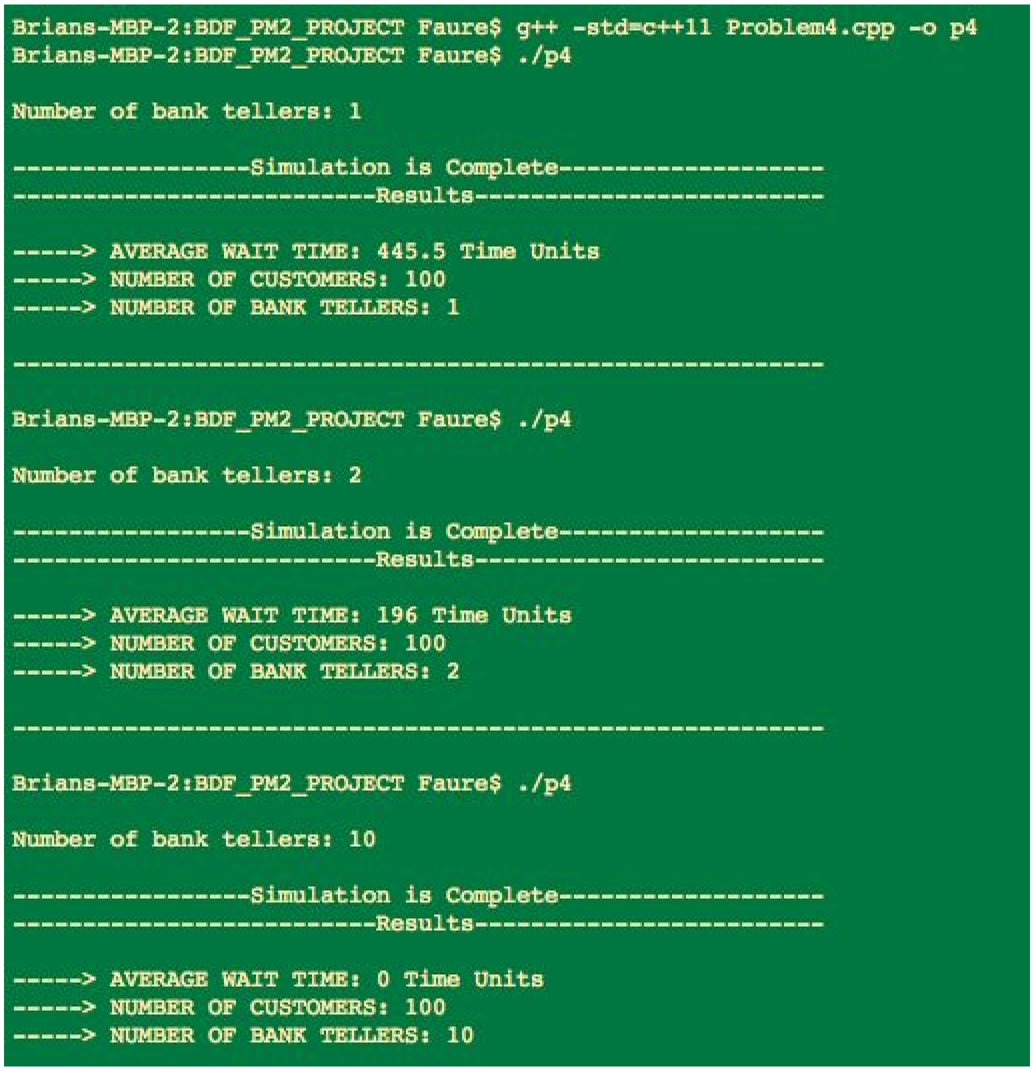
The simulation also assumes that wait time does **NOT include transaction time**

(iv.) Assume 10 customers arrive every minute for 10 minutes. Assume each service time is of duration 1 unit. What is the average wait time for the 100 customers when there is (1) 1 Teller, (ii) 2 Tellers, (iii) 10 Tellers? [1 point]

→ See Problem4.cpp file

Problem4.cpp is a copy of Problem3.cpp with some changes, the number of customers is set to 100 without the user needing to input it. I have also commented out all of the Mid-Simulation updates such that the only outputs are the request for a number of tellers along with the Post-Simulation Final output which is the same as in Problem3.cpp.

The following are the results I found when running the simulation with 1, 2, and 10 Tellers:



```
Brians-MBP-2:BDF_PM2_PROJECT Faure$ g++ -std=c++11 Problem4.cpp -o p4
Brians-MBP-2:BDF_PM2_PROJECT Faure$ ./p4

Number of bank tellers: 1

-----------------Simulation is Complete-------------------
--------------------------Results-------------------------

-----> AVERAGE WAIT TIME: 445.5 Time Units
-----> NUMBER OF CUSTOMERS: 100
-----> NUMBER OF BANK TELLERS: 1

----------------------------------------------------------

Brians-MBP-2:BDF_PM2_PROJECT Faure$ ./p4

Number of bank tellers: 2

-----------------Simulation is Complete-------------------
--------------------------Results-------------------------

-----> AVERAGE WAIT TIME: 196 Time Units
-----> NUMBER OF CUSTOMERS: 100
-----> NUMBER OF BANK TELLERS: 2

----------------------------------------------------------

Brians-MBP-2:BDF_PM2_PROJECT Faure$ ./p4

Number of bank tellers: 10

-----------------Simulation is Complete-------------------
--------------------------Results-------------------------

-----> AVERAGE WAIT TIME: 0 Time Units
-----> NUMBER OF CUSTOMERS: 100
-----> NUMBER OF BANK TELLERS: 10
```

To summarize the customer wait times:

1 Teller Average Wait Time = **445.5 Time Units**
2 Tellers Average Wait Time = **196 Time Units**
10 Tellers Average Wait Time = **0 Time Units**

[Extra credit 2 points] Instead of assuming a uniform flow of arrival, and unit time of processing, assume a Poisson distribution for inter-arrival time, and an exponential distribution for service time. Plot the mean waiting time for a range of inter-arrival times and service times.

→ I have included a file named Extra.cpp which was my first attempt at solving the extra credit problem. The way the program runs is very similar to Problem3.cpp and Problem4.cpp though I changed a lot of the functions called by the simulation() function so that they could handle different types of customer inputs and return different results.
→ I did not get a chance to finished the extra credit assignment but if you would like to look over my process you are more than welcome.