MODULE 03 PART 2

JOSEPH A. PEÑAVERDE

A. Answer the following questions (5 points each):

a. **Which teller will have the most customers at the end of the program execution?**
   - `Teller A`

b. **Which teller will have the LEAST customers at the end of the program execution?**
   - `Teller C`

c. **What made the difference between the questions above (a and b)?**
   - `Teller A Always gets the customer on each transaction counter for the reason it has the lowest time to process on each customer and teller c has the highest.`

d. **d. Will there be a significant difference if there were more customers (as represented by the list)? Why?**
   - `No, because the time that process of each teller is fixed.`

B. Modify the code so that it takes into consideration the length of transactions each customer might have. Hint: at the current example, note that ONE CUSTOMER enters the loop each time, which means it gets distributed immediately to an open teller. Because of the complication brought about by having a time component to the customer, additional checks may be done. The modified code shall be graded using the rubrics specified below

```
# each teller has their own transaction completion capability
# Arrays that will hold each teller's list of clients
tellerA = []
tellerB = []
tellerC = []
# Integer that describes each teller's speed in completing a single
# transaction. Smaller number means faster completion time
timeItTakesA = 1
timeItTakesB = 5
timeItTakesC = 10
# Initialized variable that will contain the actual running time for every
# teller
runTimeA = 0
runTimeB = 0
runTimeC = 0
# Initialization of each teller, set to false as at the start, all tellers
# have no client
hasCustA = False
hasCustB = False
hasCustC = False
# Initial customers list, where the alphabet is exploded into an array
aCustomers = list("ABCDEFGHIJKLMNOPQRSTUVWXYZ")
# Initial count of the array
curCount = len(aCustomers)
# Transaction counter (for the message)
transCounter = 0
# Loop continues while the customers list is still more than 0
while curCount > 0:
```

```
    transCounter = transCounter + 1

    if hasCustA == False:
        if len(aCustomers) > 0:
            tellerA.append(aCustomers.pop(0))
            hasCustA = True
        else:
            break

    runTimeA = runTimeA + 1
    if runTimeA == timeItTakesA:
        runTimeA = 0;
        hasCustA = False

    msgA = "Done" if runTimeA == 0 else "Busy"

    if hasCustB == False:
        if len(aCustomers) > 0:
            tellerB.append(aCustomers.pop(0))
            hasCustB = True
        else:
            break
    runTimeB = runTimeB + 1
    if runTimeB == timeItTakesB:
        runTimeB = 0;
        hasCustB = False

    msgB = "Done" if runTimeB == 0 else "Busy"

    if hasCustC == False:
        if len(aCustomers) > 0:
            tellerC.append(aCustomers.pop(0))
            hasCustC = True
        else:
            break

    runTimeC = runTimeC + 1
    if runTimeC == timeItTakesC:
        runTimeC = 0;
        hasCustC = False
    msgC = "Done" if runTimeC == 0 else "Busy"
# The next 2 lines are used for debugging and tracing purposes
transMsg = "At Round #" + str(transCounter) + " | Teller A is " + msgA + " |
Teller B is " + msgB + " | Teller C is " + msgC
print(transMsg)

print("The clients that went to teller A were: " + ", ".join(tellerA))
print("The clients that went to teller B were: " + ", ".join(tellerB))
print("The clients that went to teller C were: " + ", ".join(tellerC))
```