DATA 7202 Assignment 4 Report

Name: Yupeng Wu
No: 45960600

1.

(a).

$\hat{\theta} \approx -0.0713$

(b).

The 95% confidence interval for $\hat{\theta}$ is about $(-0.2746, 0.132)$.

The confidence interval shows that the absolute value of $\hat{\theta}$ is larger than 0.20, which means it does not meet the FDA requirement for bioequivalence. The new drug is not bioequivalent.

2.

(a).

Our objective is to carry out DES simulation study for the M/M/c queuing system and make sure that 90% of the customers' waiting time is no more than 8 minutes.

The arrivals are governed by a Poisson process, there are c desks used to serve customers and the service times are exponentially distributed.

To set the stage, we consider the following facts.

The inter-arrival times are exponentially distributed with mean $\frac{1}{\lambda}$ and the service time is exponentially distributed with mean $\frac{1}{\mu}$.

The server utilization $\rho = \frac{\lambda}{c\mu} < 1$ then the system has a stationary distribution.

The probability that an arriving customer is forced to join the queue is given by:

$$C\left(c, \frac{\lambda}{\mu}\right) = \frac{1}{1 + (1-\rho)\left(\frac{c!}{(c\rho)^c}\right)\sum_{k=0}^{c-1}\frac{(c\rho)^k}{k!}}$$

The average number of customers in the system is given by:

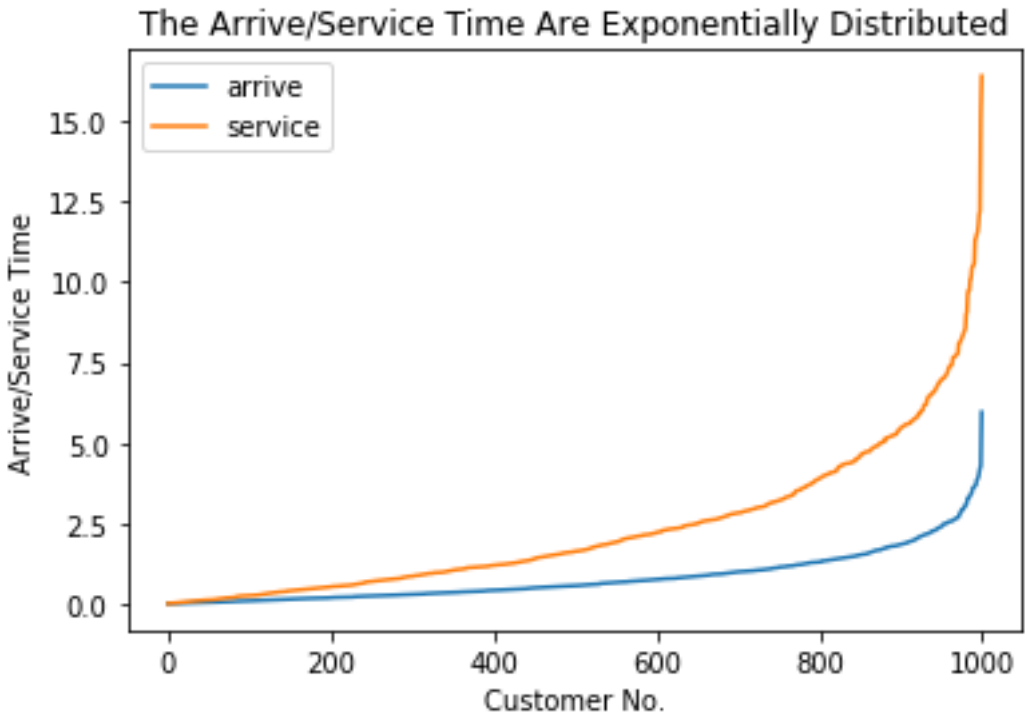
$$\frac{\rho}{1-\rho}C\left(c, \frac{\lambda}{\mu}\right) + c\rho$$

The average response time, which is the average of total amount of a time a customer spends in the queue and in service is given by:

$$\frac{C(c, \frac{\lambda}{\mu})}{c\mu - \lambda} + \frac{1}{\mu}$$

(b).

From the given dataset, we can plot the two columns after sort them:

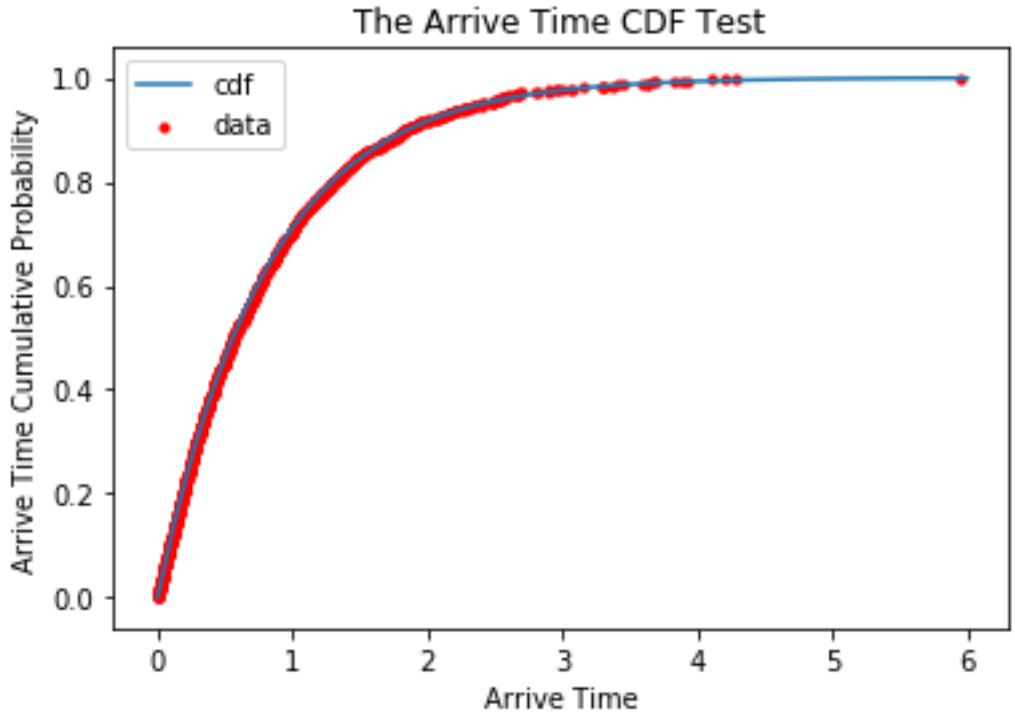


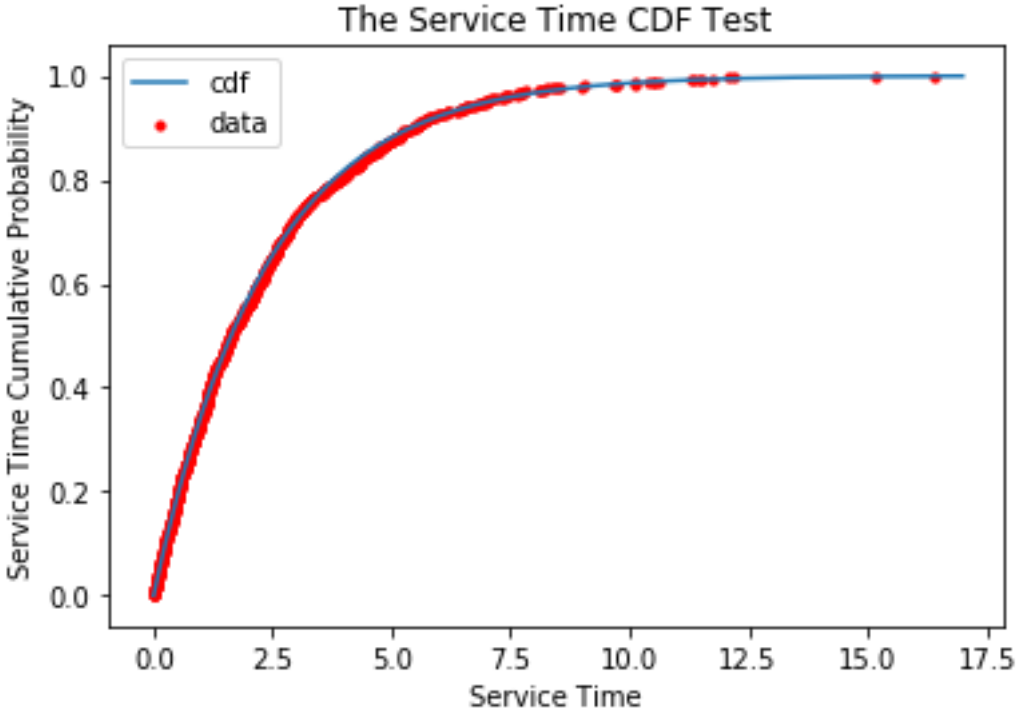
And we can see from the diagram that the inter-arrival time and the service time are both exponentially distributed. Then we can use 'expon.fit' function to compute their mean. Here we have:

$$\frac{1}{\lambda} \approx 0.803, \qquad \frac{1}{\mu} \approx 2.347$$

$$\lambda \approx 1.245, \qquad \mu \approx 0.426, \qquad \rho = \frac{\lambda}{c\mu} \approx \frac{2.92}{c} < 1$$

After getting the λ and the μ, we need to do a cumulative distribution function CDF test to show whether the data has such exponential distribution.

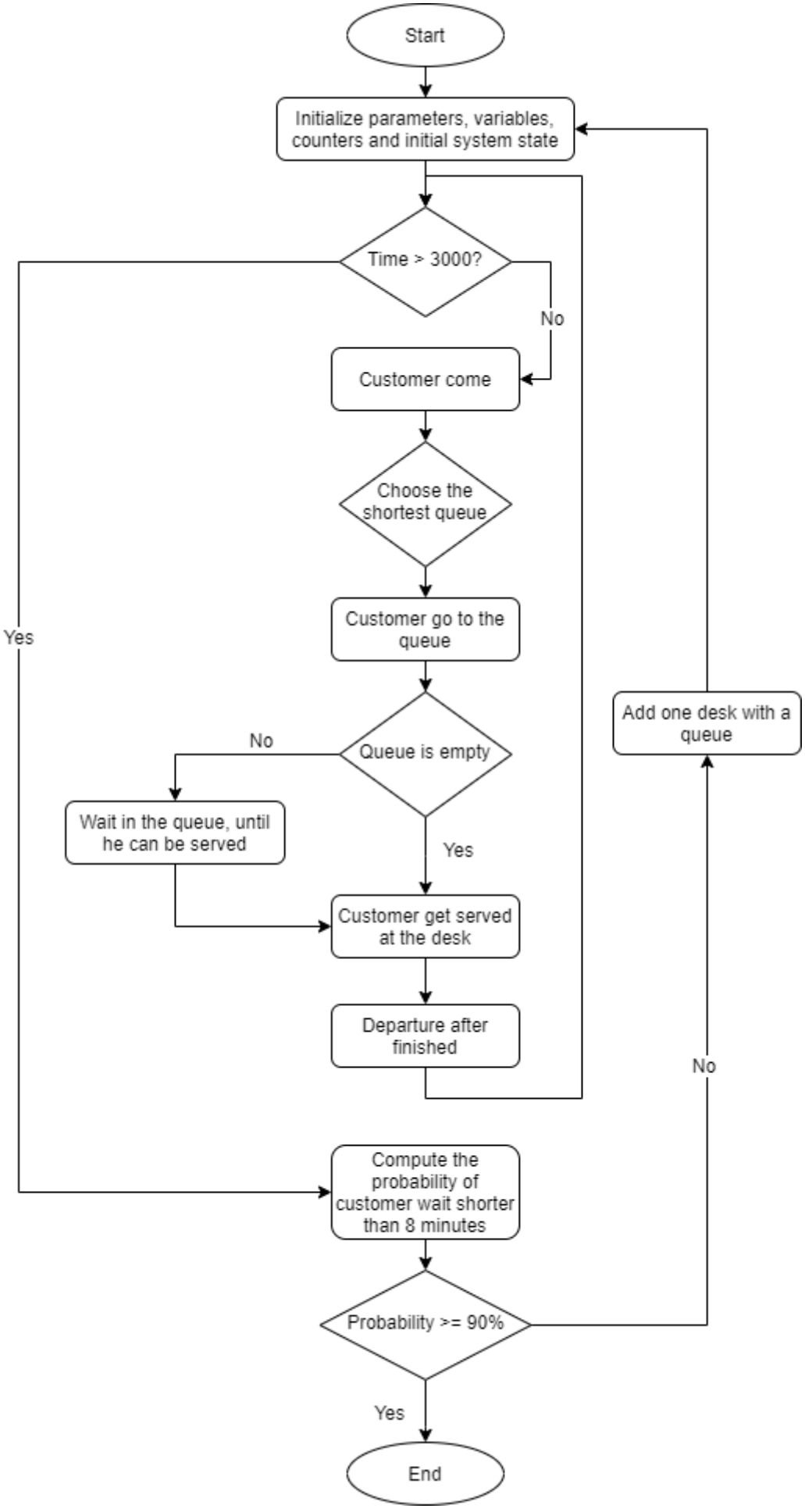


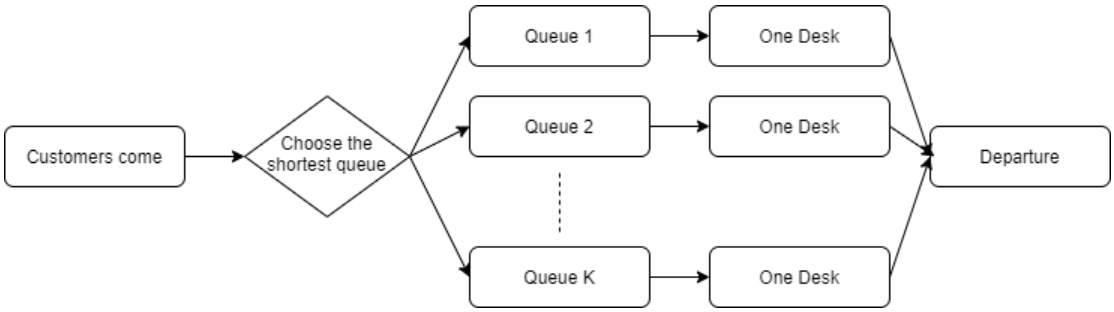


From the two plots we can see that all the data points (red) are well located on the distribution (blue). So, the values of $\lambda \approx 1.245$ and $\mu \approx 0.426$ are correct.

In order to meet the condition of model stability, the number of desks should bigger than 2.91. So, we started with initially set there are 3 desks in the system to find out what is the percentage of waiting less than 8 minutes.

The diagrams that describe the dynamics are:


Start
Initialize parameters, variables, counters and initial system state
Time > 3000?
No
Yes
Customer come
Choose the shortest queue
Customer go to the queue
Queue is empty
No
Yes
Wait in the queue, until he can be served
Customer get served at the desk
Departure after finished
Add one desk with a queue
No
Compute the probability of customer wait shorter than 8 minutes
Probability >= 90%
Yes
End



(c).

Here we show the table of the probabilities of customers wait no longer than 8 minutes in a waiting before they are served and the corresponding number of desks c. Since we need to ensure the stability of the system, we started with 3 desks in the system.

Number of desks c	3	4	5	6
Probability	18.544%	94.844%	97.359%	98.256%
95% CI	(41.384 %, 44.543 %)	(94.143%, 95.545%)	(99.093%, 99.603%)	(99.793%, 99.998%)

Here we can see that the minimally number of available desks to meet the service requirements is 4. The corresponding confidence interval is (94.143%, 95.545%). For more information, please check the following table.

Minimally number of available desks c	4
Server utilization ρ	0.73
Probability of waiting less than 8 minutes	94.844%
95% CI for the probability	(94.143%, 95.545%)
Probability of customer is forced to join the queue	47.68%
Average number of customers in system	4.212
Response time	3.384

(d).

In my opinion, the minimum number of the desks in the system is 4 and the probability of the customers' waiting time is less than 8 minutes is 94.844%. Besides, this probability increases a lot from 3 desks to 4 desks. When there are more desks in the system, the average waiting time become smaller. Besides, we should always keep the number of the desks larger than 2.91 in order to keep the stability of the system.

(e). Appendix

I used 'simpy' for the simulation. First, the 'simpy.Environment()' build the background for the system. The 'source' function generates a system in the environment. The desk are generated in the environment by the 'simpy.Resource' function. When the environment's current time is less than 3000 units of time, a source will generate customers randomly. The shortest queue is also picked while the customer is generated. The 'customer' function is used to set the information for a particular customer, and the waiting time of the customer is calculated in this function.

```
# -*- coding: utf-8 -*-
"""
Created on Tue Jun  9 15:55:25 2020

@author: Yupeng Wu
"""

from scipy.stats import expon
import numpy as np
import matplotlib.pyplot as plt

# Get the data from csv
def readData(filename):
    data = []
    with open(filename) as f:
        for line in f:
         data.append(line.strip().split(','))
    return data
# Expon pdf
def func(x, a, b, c):
    return a * np.exp(-b * x) + c

# Prepare data
data = readData('data.csv')[1:]
Customers = range(len(data))
arrive = [0 for i in Customers]
service = [0 for i in Customers]

for i in Customers:
    arrive[i] = float(data[i][0])
    service[i] = float(data[i][1])

# Generate plot to show the data is exponentially distributed
plt.plot(sorted(arrive), label = 'arrive')
plt.plot(sorted(service), label = 'service')
plt.axis()
plt.xlabel('Customer No.')
plt.ylabel('Arrive/Service Time')
plt.title('The Arrive/Service Time Are Exponentially Distributed')
plt.legend()
plt.show()

# Compute the mean of the exponential distribution
# scale_arrive = 0.80341291196 (mean arrive time: 1/lambda)
loc_arrive, scale_arrive = expon.fit(arrive, floc=0)
# scale_service = 2.3468397434240003 (mean service time: 1/mu)
loc_service, scale_service = expon.fit(service, floc=0)

# CDF test whether they are exponential
exp_arrive = expon.cdf(np.linspace(0,6,len(arrive)), scale=scale_arrive)
plt.plot(np.linspace(0,6,len(arrive)), exp_arrive, label='cdf')
plt.scatter(sorted(arrive), np.arange(len(arrive))/len(arrive), s=10, c='red', label='data')
plt.axis()
plt.xlabel('Arrive Time')
plt.ylabel('Arrive Time Cumulative Probability')
plt.title('The Arrive Time CDF Test')
plt.legend()
plt.show()

exp_service = expon.cdf(np.linspace(0,17,len(service)), scale=scale_service)
plt.plot(np.linspace(0,17,len(service)), exp_service, label='cdf')
plt.scatter(sorted(service), np.arange(len(service))/len(service), s=10, c='red', label='data')
plt.axis()
plt.xlabel('Service Time')
plt.ylabel('Service Time Cumulative Probability')
plt.title('The Service Time CDF Test')
plt.legend()
plt.show()


import random
import simpy

RANDOM_SEED = 12345
# Total number of customers (set high to make sure there are enough customers)
NEW_CUSTOMERS = 10000
# The scale of the twp exponential distributions
# Arrival 1.245 customer per unit time on average
lambda_arrive = 1.0 / 0.80341291196
```

```
# Service 0.426 customer per unit time on average
mu_service = 1.0 / 2.3468397434240003
# Generate new customers roughly every lambda_arrive seconds
INTERVAL_CUSTOMERS = lambda_arrive


def source(env, number, interval, counters):
    """Source generates customers randomly"""
    for i in range(number):
        lengths = []
        for rescource in counters:
            lengths.append(len(rescource.queue) + rescource.count)
        desk = lengths.index(min(lengths))
        counter = counters[desk]
        # Generate customer with the service time
        c = customer(env, 'Customer%02d' % i, counter, desk, service= mu_service)
        env.process(c)
        t = random.expovariate(interval)
        yield env.timeout(t)


def customer(env, name, counter, desk, service):
    """Customer arrives, is served and leaves."""
    arrive = env.now
    #print('%7.4f %s: Here I am to desk %s' % (arrive, name, desk))

    with counter.request() as req:
        # Wait for the counter
        yield req

        wait = env.now - arrive

        WAIT.append(wait)

        # We got to the counter
        #print('%7.4f %s: Waited %6.3f' % (env.now, name, wait))

        tib = random.expovariate(service)
        yield env.timeout(tib)
        #print('%7.4f %s: Finished' % (env.now, name))


# Setup and start the simulation
print('Air Secure Service')
random.seed(RANDOM_SEED)
# Generate simpy environment
env = simpy.Environment()
# Set the waiting time list
WAIT = []

# Start processes and run
# Set 4 queues, where each queue has 1 desk
c = 6
counters = [simpy.Resource(env, capacity=1) for i in range(c)]
env.process(source(env, NEW_CUSTOMERS, INTERVAL_CUSTOMERS, counters))
# 3000 units of times
env.run(until=3000)

# Compute the percentage
# Generate zero-one list, if waiting time less than 8 is 1, else 0
WAIT_8 = []
for time in WAIT:
    if time <= 8:
        WAIT_8.append(1)
    else:
        WAIT_8.append(0)

mean_8 = np.mean(WAIT_8)
std_8 = np.std(WAIT_8)/np.sqrt(len(WAIT_8))

rho = lambda_arrive/(c*mu_service)
print('\n Server utilization rho = ', round(rho, 3), ' < 1, with', c, 'desks/servers.')

print('\n The system will have a stationary distribution if the desk is more than: \n',
      round(lambda_arrive/mu_service, 3))

print('\n The probability of customers waiting less than 8 minutes: \n',
      round(mean_8*100, 3), '%')
```

```
print('\n 95% CI for probability of customers waiting less than 8 minutes: \n (',
      round((mean_8 - 1.96*std_8)*100, 3), '%,', round((mean_8 + 1.96*std_8)*100, 3), '% )')

def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return (n*factorial(n-1))

C_prob = 1/(1+(1-rho)*(factorial(c)/((c*rho)**c))*sum(((c*rho)**k)/factorial(k)
                                                      for k in range(c)))
print('\n The probability that an arriving customer (all servers are occupied) is forced to join the queue is: \n',
      round(C_prob*100, 3), '%')

C_sys = rho/(1-rho)*C_prob + c*rho
print('\n The average number of customers in the system (in service and in the queue) is: \n',
      round(C_sys, 3))

C_response = C_prob/(c*mu_service-lambda_arrive) + 1.0/mu_service
print('\n The response time (the total amount of time a customer spends in both the queue and in service) is: \n',
      round(C_response, 3))
```