BHao_Assign2

Problem 2.6.1

```
interarrival = 1.25  # minutes
lambda = 1 / interarrival  # entities per minute
lambda
```

```
## [1] 0.8
```

```
S = 1  # minutes
mu = 1 / S  # entities per minute
mu
```

```
## [1] 1
```

```
rho = lambda / mu  # utilization
rho
```

```
## [1] 0.8
```

```
L = rho / (1 - rho)  # steady state average number of entities in the system for c = 1 and M/M/1
L
```

```
## [1] 4
```

```
W = L / lambda  # steady state average time in system in minutes
W
```

```
## [1] 5
```

```
Wq = W - S  # steady state average time in queue in minutes
Wq
```

```
## [1] 4
```

```
Lq = lambda * Wq  # steady state average number of entities in queue
Lq
```

```
## [1] 3.2
```

Problem 2.6.2

```
interarrival = 1.25  # minutes
lambda = 1 / interarrival  # entities per minute
lambda
```

```
## [1] 0.8
```

```
a = 0.1  # minutes
b = 1.9  # minutes
S = (a + b) / 2  # minutes
S
```

```
## [1] 1
```

```
sigma = sqrt((b - a)^2 / 12)
sigma
```

```
## [1] 0.5196152
```

```
mu = 1 / S  # entities per minute
mu
```

```
## [1] 1
```

```
rho = lambda / mu  # utilization
rho
```

```
## [1] 0.8
```

```
Wq = (lambda * (sigma^2 + 1 / mu^2)) / (2 * (1 - lambda / mu))  # steady state average time in queue in
Wq
```

```
## [1] 2.54
```

```
W = Wq + S  # steady state average time in system in minutes
W
```

```
## [1] 3.54
```

```
L = lambda * W  # steady state average number of entities in system
L
```

```
## [1] 2.832
```

```
Lq = lambda * Wq  # steady state average number of entities in queue
Lq
```

```
## [1] 2.032
```

- Wq: decreases to 2.54 minutes from 4 minutes
- W: decreases to 3.54 minutes from 5 minutes
- Lq: decreases to 2.03 entities from 3.2 entities
- L: decreases to 2.83 entities from 4 entities
- rho: remains unchanged since lambda and mu remained unchanged

The uniform distribution of service times between 0.1-1.9 minutes results in lower in system and in queue wait times and entities because it eliminates the longer servicing times in the tail of the exponential distribution, which drag up the average servicing time

Problem 2.6.3

```
interarrival = 1.25  # minutes
lambda = 1 / interarrival  # entities per minute
lambda
```

```
## [1] 0.8
```

```
a = 0.1  # minutes
b = 1.9  # minutes
m = 1.0  # minutes
S = (a + m + b) / 3  # minutes
S
```

```
## [1] 1
```

```
sigma = sqrt((a^2 + m^2 + b^2 - a*m - a*b -b*m) / 18)
sigma
```

```
## [1] 0.3674235
```

```
mu = 1 / S  # entities per minute
mu
```

```
## [1] 1
```

```
rho = lambda / mu  # utilization
rho
```

```
## [1] 0.8
```

```
Wq = (lambda * (sigma^2 + 1 / mu^2)) / (2 * (1 - lambda / mu))  # steady state average time in queue in
Wq
```

```
## [1] 2.27
```

```
W = Wq + S  # steady state average time in system in minutes
W
```

```
## [1] 3.27
```

```
L = lambda * W  # steady state average number of entities in system
L
```

```
## [1] 2.616
```

```
Lq = lambda * Wq  # steady state average number of entities in queue
Lq
```

```
## [1] 1.816
```

- Wq: decreases to 2.27 minutes from 2.54 minutes and from 4 minutes vs. the uniform distribution of service times and the exponential distribution
- W: decreases to 3.27 minutes from 3.54 minutes and from 5 minutes
- Lq: decreases to 1.82 entities from 2.03 entities and 3.2 entities
- L: decreases to 2.62 entities from 2.83 entities and 4 entities
- rho: remains unchanged since lambda and mu remained unchanged

In this case, the triangular distribution of service times between 0.1-1.9 minutes with a mode of 1.0 minutes resulted in a lower standard deviation of service times, which in turn drove down in system and in queue wait times and entity averages

Problem 2.6.5

```
c = 3  # servers
interarrival = 1.25  # minutes
lambda = 1 / interarrival  # entities per minute
lambda
```

```
## [1] 0.8
```

```
S = 3  # minutes
mu = 1 / S  # entities per minute
mu
```

```
## [1] 0.3333333
```

```
rho = lambda / (c * mu)  # utilization
rho
```

```
## [1] 0.8
```

```
sum_n_c = 0
for (n in 0:(c-1)) {
  sum_n_c = sum_n_c + (c * rho)^n / factorial(n)
}
p0 = 1 / ((c*rho)^c / (factorial(c) * (1 - rho)) + sum_n_c)
Lq = (rho * (c * rho)^c * p0) / (factorial(c) * (1 - rho)^2) # steady state average number of entities
Lq
```

```
## [1] 2.588764
```

```
L = Lq + lambda / mu  # steady state average number of entities in the system for c = 1 and M/M/1
L
```

```
## [1] 4.988764
```

```
W = L / lambda  # steady state average time in system in minutes
W
```

```
## [1] 6.235955
```

```
Wq = W - S  # steady state average time in queue in minutes
Wq
```

```
## [1] 3.235955
```

- Wq: decreases to 3.24 minutes from 4 minutes
- W: increases to 6.24 minutes from 5 minutes
- Lq: decreases to 2.59 entities from 3.2 entities
- L: increases to 5.00 entities from 4 entities
- rho: remains unchanged since lambda and mu remained unchanged

In this case, in queue wait times and entity averages decrease, but in system wait times and entity averages increase. I assume this is due to the increased servicing time and the much longer servicing times in the tail of the distribution.

Problem 2.6.12

```
rho_calc = function(prob, lambda, c, mu) {
  (prob * lambda) / (c * mu)
}

interarrival = 6  # minutes
lambda = 1 / interarrival  # entities per minute
```

```r
# Sign in
rho_calc(1, lambda, 2, 1/3)
```

```
## [1] 0.25
```

```r
# Registration
rho_calc(0.9, lambda, 1, 1/5)
```

```
## [1] 0.75
```

```r
# Trauma rooms
rho_calc(0.1, lambda, 2, 1/90)
```

```
## [1] 0.75
```

```r
# Exam rooms
rho_calc(0.9, lambda, 3, 1/16)
```

```
## [1] 0.8
```

```r
# Treatment rooms
rho_calc(0.9*0.6+0.1, lambda, 2, 1/15)
```

```
## [1] 0.8
```

Yes, this clinic will work because the utilization or local traffic intensity at each node is less than 1. If an additional server were available, it should be added to either exam rooms or treatment rooms as they have the highest current utilization rates. Given that an additional server would reduce the treatment room utilization rate more than it would reduce that for the exam room, I would allocate the first additional server to the treatment rooms.

```r
# Exam rooms
rho_calc(0.9, lambda, 4, 1/16)
```

```
## [1] 0.6
```

```r
# Treatment rooms
rho_calc(0.9*0.6+0.1, lambda, 3, 1/15)
```

```
## [1] 0.5333333
```