Q1.1

For a M/M/1 system, average time spent by a passenger is:

$$T = \frac{\overline{N}}{\lambda} = \frac{1}{\mu - \lambda}$$

The system occupancy is:

$$\overline{N} = \frac{\rho}{1-\rho} = \frac{\lambda}{\mu - \lambda}$$

Q1.2

The relationship between total time spent in the system (T), queueing delay (W) and service time ($\bar{x}$) is:

$$T = W + \bar{x}$$

The relationship between the number of customers (packets) in the queue ($\overline{N_q}$), that are currently serviced ($\overline{N_s}$), and the total system occupancy ($\overline{N}$) is:

$$\overline{N} = \overline{N_s} + \overline{N_q}$$

Q1.3

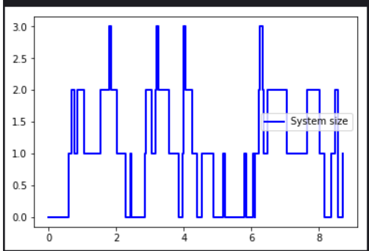
The simulation result for M/M/1 are as below:

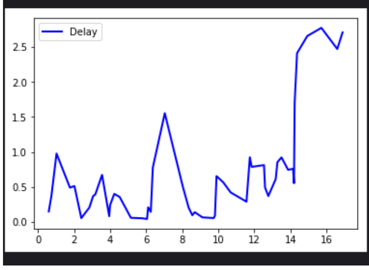
```
Mean theoretical delay: 1.000000

Mean theoretical size : 3.000000

Mean practical delay: 0.988824

Mean practical size : 2.970748
```





Q1.4

We compare for maxsteps = 100, and maxsteps = 10^6, and we found that when steps is 10^6 the practical values are close to theoretical values.

maxsteps = 1000000

```
Mean theoretical delay: 1.000000

Mean theoretical size : 3.000000

Mean practical delay: 1.008329

Mean practical size : 3.019278
```

maxsteps = 100

```
Mean theoretical delay: 1.000000

Mean theoretical size : 3.000000

Mean practical delay: 1.265213

Mean practical size : 4.003276
```

Q1.5

The simulated results for M/D/1, D/M/1 and D/D/1 system:

M/D/1 D/M/1 D/D/1

Q1.6

The codes for part 1 are as below:

The theoretical calculation:

```
def Theoreticalmm1(srate, arate):
    '''
    Theoretically calculates and prints mean delay and system size.

    Inputs: service and arrival rates.
    '''

    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''
    meandelay = 1 / ( srate - arate ) #calculate average delay T
    p =  arate / srate
    meansize = p / (1 - p) #calculate system occupancy N

    # Hint for display...
    print('Mean theoretical delay: {:4f} \n'.format(meandelay))
    print('Mean theoretical size : {:4f} \n'.format(meansize))
```

- M/M/1

```
maxsteps = 100000 # simulation steps
srate = 6  # service rate
arate = 5  # arrival rate


# create simulation
simulation = DESmm1(srate, arate, maxsteps)

# main loop
for i in range(maxsteps):
    intarrive = np.random.exponential(1/arate)
    simulation.packetarrival(intarrive)
    servetime = np.random.exponential(1/srate)
    simulation.nextstep(servetime)

# calculate and print theoretical values
# you can also do this in Matlab if you prefer!
Theoreticalmm1(srate, arate)

# calculate and print practical delay, size values
# optionally visualise
simulation.practicalcalc(True, True)
```

- D/M/1

```python
maxsteps = 100000 # simulation steps
srate = 6  # service rate
arate = 5  # arrival rate


# create simulation
simulation = DESmm1(srate, arate, maxsteps)

# main loop
for i in range(maxsteps):
    intarrive = 1/arate
    simulation.packetarrival(intarrive)
    servetime = np.random.exponential(1/srate)
    simulation.nextstep(servetime)

# calculate and print theoretical values
# you can also do this in Matlab if you prefer!
Theoreticalmm1(srate, arate)

# calculate and print practical delay, size values
# optionally visualise
simulation.practicalcalc(True, True)
```

- M/D/1

```python
maxsteps = 100000 # simulation steps
srate = 6  # service rate
arate = 5  # arrival rate


# create simulation
simulation = DESmm1(srate, arate, maxsteps)

# main loop
for i in range(maxsteps):
    intarrive = np.random.exponential(1/arate)
    simulation.packetarrival(intarrive)
    servetime = 1/srate
    simulation.nextstep(servetime)

# calculate and print theoretical values
# you can also do this in Matlab if you prefer!
Theoreticalmm1(srate, arate)

# calculate and print practical delay, size values
# optionally visualise
simulation.practicalcalc(True, True)
```

- D/D/1

```python
maxsteps = 100000 # simulation steps
srate = 6  # service rate
arate = 5  # arrival rate


# create simulation
simulation = DESmm1(srate, arate, maxsteps)

# main loop
for i in range(maxsteps):
    intarrive = 1/arate
    simulation.packetarrival(intarrive)
    servetime = 1/srate
    simulation.nextstep(servetime)

# calculate and print theoretical values
# you can also do this in Matlab if you prefer!
Theoreticalmm1(srate, arate)

# calculate and print practical delay, size values
# optionally visualise
simulation.practicalcalc(True, True)
```

Q2.1

Code and simulation results of M/M/2

```python
def Theoreticalmmk(srate, arate, k):
    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''

    p = arate/(k*srate)

    p0_1 = 0.0

    for i in range (k):
        p0_1 += ((k*p)**i)/(math.factorial(i))
    p0_2 = ((k*p)**k)*(1/(1-p))/(math.factorial(k))

    p0 = (p0_1 + p0_2)**(-1)

    Nq = (((k*p)**k)*p)*p0/(math.factorial(k)*((1-p)**2))
    W = Nq/arate
    meandelay = 1/srate + W
    meansize = arate * meandelay

    print('Mean theoretical delay: {:4f} \n'.format(meandelay))
    print('Mean theoretical size : {:4f} \n'.format(meansize))
```

```python
maxsteps = 100000  # simulation steps
srate = 3 # service rate
arate = 5  # arrrival rate
nbrservers = 2  # number of servers

simulation = DESmmc(srate, arate, nbrservers, maxsteps)

for i in range(maxsteps):
    intarrive = np.random.exponential(1/arate)  # interarrival time
    simulation.packetarrival(simulation.Q, intarrive)
    servetime = np.random.exponential(1/srate)  # service time
    simulation.nextstep(servetime)

Theoreticalmmk(srate, arate, nbrservers)

simulation.practicalcalc()
```

M/M/2

```
Mean theoretical delay: 1.090909

Mean theoretical size : 5.454545

Mean practical delay: 1.125470

Mean practical size : 5.594555
```

Bonus Question:

We compare when srate = 3, arate = 5 (stable) and srate = 2, arate = 5(unstable).

M/D/2 (stable)

```
Mean practical delay: 0.716820

Mean practical size : 3.574348
```

M/D/2 (unstable)

```
Mean practical delay: 1381.523406

Mean practical size : 5517.220305
```

D/M/2 (stable)

```
Mean practical delay: 0.666409

Mean practical size : 3.332057
```

D/M/2 (stable)

```
Mean practical delay: 1376.311357

Mean practical size : 5510.904563
```

D/D/2 (stable)

```
Mean practical delay: 0.333333

Mean practical size : 1.666653
```

D/D/2 (unstable)

```
Mean practical delay: 1389.350000

Mean practical size : 5557.499982
```

Q3.1

1. M/M/1 （srate = 6， arate = 5）

```
Mean theoretical delay: 1.000000

Mean theoretical size : 5.000000
```

2. Parallel M/M/1

```
def Theoreticalmm_parallel(srate, arate,parallel):
    '''
    Theoretically calculates and prints mean delay and system size.

    Inputs: service and arrival rates.
    '''

    '''
    ??? 'You can optionally do this in Matlab, if you wish!'
    '''
    meandelay = 1 / (srate-arate/2) #calculate average delay T
    p =  (arate/2) / srate
    meansize = parallel*p / (1-p) #calculate system occupancy N

    # Hint for display...
    print('Mean theoretical delay: {:4f} \n'.format(meandelay))
    print('Mean theoretical size : {:4f} \n'.format(meansize))
```

```
Mean theoretical delay: 2.000000

Mean theoretical size : 10.000000
```

3. M/M/2 （srate = 3， arate = 5）

```
Mean theoretical delay: 1.090909

Mean theoretical size : 5.454545
```

Q3.2

Simulation results of two parallel M/M/1 system are as below:

```
Q0 mean practical delay: 2.457435

Q0 mean practical size : 6.219832

Q1 mean practical delay: 1.907304

Q1 mean practical size : 4.788013

Combined mean practical delay: 2.182370

Combined mean practical size : 11.007844
```

Q3.3

The 1st M/M/1 system:

$$T = \frac{1}{\mu - \lambda} = \frac{1}{6-5} = 1$$

$$N = \lambda T = 5$$

The 2nd two parallel M/M/1 system:

$$\mu' = 3, \qquad \lambda' = 0.5 \times \lambda = 2.5$$

$$T = \frac{1}{\mu' - \lambda'} = \frac{1}{3-2.5} = 2$$

$$N = 2 \times \lambda' T = 2 \times 2.5 \times 2 = 10$$

The 3rd M/M/2 system:

$$p_0 = \left[\sum_{k=0}^{m-1} \frac{(m\rho)^k}{k!} + \left(\frac{(m\rho)^m}{m!}\right)\left(\frac{1}{1-\rho}\right) \right]^{-1}$$

$$\bar{N}_q = \left(\frac{(m\rho)^m \rho}{2(1-\rho)^2}\right) p_0$$

$$W = \frac{\bar{N}_q}{\lambda}$$

$$T = W + \frac{1}{\mu} = \frac{12}{11} = 1.0909$$

$$N = \lambda T = 5 \times \frac{12}{11} = \frac{60}{11} = 5.4545$$

The results we calculated are the same as the theorical results.

Q3.4

From the simulated and calculated results, we can see that the first method has the smallest delay and smallest system size, but practically if we consider about the cost, method 3 is the best method. Because it also has a good performance that is close to method 1 and it also cost less since fast router can be expensive. In conclusion, by taking cost and performance into account, we think method 3 is the best one.