In [2]:
```
from collections import deque
import numpy as np
from scipy.special import comb
import random
import matplotlib.pyplot as plt

experimentCount = 10**6
arrivalRate =  [0.2, 0.3, 0.4, 0.45, 0.49, 0.495];
departureRate = 0.5

dict={}
queueDelay = []

#total jobs when arrival rate = 0.45
jobs045=0

for i in range(len(arrivalRate)):
    delay=0
    servers_jobs = [0, 0, 0, 0, 0]
    arrRate = arrivalRate[i]

    #initial 5 servers
    queue1 = deque()
    queue2 = deque()
    queue3 = deque()
    queue4 = deque()
    queue5 = deque()
    for j in range(experimentCount):
        job_number=0

        #find numbers of job arrival
        for job in range(5):
            rate = random.random()
            if(rate<arrRate):
                job_number+=1
                if(arrRate==0.45):
                    jobs045+=1

        #add jobs to shortest queue
        for k in range(job_number):
            if(servers_jobs.index(min(servers_jobs)) == 0):
                queue1.append(j)
                servers_jobs[0] = len(queue1)
            elif(servers_jobs.index(min(servers_jobs)) == 1):
                queue2.append(j)
                servers_jobs[1] = len(queue2)
            elif(servers_jobs.index(min(servers_jobs)) == 2):
                queue3.append(j)
                servers_jobs[2] = len(queue3)
            elif(servers_jobs.index(min(servers_jobs)) == 3):
                queue4.append(j)
```

```
            servers_jobs[3] = len(queue4)
        elif(servers_jobs.index(min(servers_jobs)) == 4):
            queue5.append(j)
            servers_jobs[4] = len(queue5)


    #check if job finished, release server
    if(random.random() < departureRate and servers_jobs[0]>0):
            out = j - queue1.popleft()
            delay += out
            if(arrRate == 0.45):
                if(dict.get(out)):
                    dict[out] += 1
                else:
                    dict[out] = 1
            servers_jobs[0] = len(queue1)
    if(random.random() < departureRate and servers_jobs[1] > 0):
            out = j - queue2.popleft()
            delay += out
            if(arrRate == 0.45):
                if(dict.get(out)):
                    dict[out] += 1
                else:
                    dict[out] = 1
            servers_jobs[1] = len(queue2)
    if(random.random() < departureRate and servers_jobs[2] > 0):
            out = j - queue3.popleft()
            delay += out
            if(arrRate == 0.45):
                if(dict.get(out)):
                    dict[out] += 1
                else:
                    dict[out] = 1
            servers_jobs[2] = len(queue3)
    if(random.random() < departureRate and servers_jobs[3] > 0):
            out = j - queue4.popleft()
            delay += out
            if(arrRate == 0.45):
                if(dict.get(out)):
                    dict[out] += 1
                else:
                    dict[out] = 1
            servers_jobs[3] = len(queue4)
    if(random.random() < departureRate and servers_jobs[4] > 0):
            out = j - queue5.popleft()
            delay += out
            if(arrRate == 0.45):
                if(dict.get(out)):
                    dict[out] += 1
                else:
                    dict[out] = 1
            servers_jobs[4] = len(queue5)
print(delay/experimentCount)
```

```
    queueDelay.append(delay/experimentCount)

xBar=[]
yBar=[]
for key in dict.keys():
    xBar.append(key)
for val in dict.values():
    yBar.append(val)

plt.bar(np.array(xBar), np.array(yBar)/jobs045)
plt.xlabel('Possible Delay')
plt.ylabel('Number of Jobs')
plt.title('Q2: Arrival Rate : 0.45')
plt.show()
plt.plot(arrivalRate, queueDelay)
plt.xlabel('Arrival Rate')
plt.ylabel('Average Delay')
plt.title('Q1')
plt.show()
```

```
1.026128
1.701458
3.203961
5.802307
26.000409
49.822267
```