Lakshay Sharma

Hindu College

Section B

1.Write a Program to determine EOQ using various inventory models.

In [14]:
```
import cmath
import math

#EOQ using basic model
def eoq_basic(demand, setup_cost, holding_cost):
    eoq = math.sqrt((2 * demand * setup_cost) / holding_cost)
    return eoq

#EOQ using extended model with shortage
def eoq_shortage(demand, setup_cost, holding_cost, shortage_cost):
    eoq = math.sqrt((2 * demand * setup_cost) / (holding_cost + (shortage_cost / demand)))
    return eoq

#EOQ using production model
def eoq_production(demand, setup_cost, holding_cost, production_rate):
    eoq = cmath.sqrt((2 * demand * setup_cost) /(holding_cost * (1 - (demand / production_rate))))
    return eoq



def main():
    demand = int(input("Enter the demand: "))
    setup = int(input("Enter the setup cost: "))
    holding = int(input("Enter the holding cost: "))
```

```
    shortage = int(input("Enter the shortage cost: "))
    production = int(input("Enter the production rate: "))


    print("EOQ using basic model:",eoq_basic(demand, setup, holding))
    print("EOQ using extended model with shortage:",eoq_shortage(demand, setup, holding, shortage))
    print("EOQ using production model:",eoq_production(demand, setup, holding, production))


main()
```

```
Enter the demand: 300
Enter the setup cost: 30
Enter the holding cost: 22
Enter the shortage cost: 10
Enter the production rate: 320
EOQ using basic model: 28.60387767736777
EOQ using extended model with shortage: 28.582232666566636
EOQ using production model: (114.41551070947108+0j)
```

2. Write a Program to determine different characteristics using various Queuing models

In [12]:

```
import math

def mm1(l, m):
    rho = l / m
    print("Probability that the server is busy: ", rho)
    p0=1-rho
    print("Probability that the server is idle: ",p0)
    L=rho/(1-rho)
    print("Expected number of customers in the system: ",L)
    Lq= (rho**2)/(1-rho)
    print("Expected number of customers in the queue: ", Lq)
    w=1/(m-l)
    print("Average waiting time in the system: ",w)
    wq = rho/(m - l)
    print("Average waiting time in the queue: ",wq)
    return

def mmc(l,m, c):
```

```python
    rho = l/(c * m)
    print("Probability that the server is busy: ", rho)
    p0=1-rho
    print("Probability that the server is idle: ",p0)
    r=rho**c
    L = r/((math.factorial(c))*p0)*(((c*m)/l)-1+r/p0)
    print("Expected number of customers in the system: ",L)
    Lq = r / ((math.factorial(c))*p0)*(rho/(p0 ** 2))*((c * m / l) - 1 - rho ** c + (rho ** (c + 1)) / p0)
    print("Expected number of customers in the queue: ", Lq)
    W = L / (l * (1 - (L / c)))
    print("Average waiting time in the system: ",W)
    Wq = Lq / (l * (1 - (L / c)))
    print("Average waiting time in the queue: ",Wq)
    return

# M/G/1 queue
def mg1(l, m):
    st=1/m
    rho = l * st
    print("Probability that the server is busy: ", rho)
    p0=1-rho
    print("Probability that the server is idle: ",p0)
    Lq = ((l ** 2) * (st ** 2)) / (2 * p0)
    print("Expected number of customers in the queue: ", Lq)
    L = Lq + rho
    print("Expected number of customers in the system: ",L)
    Wq = Lq / l
    print("Average waiting time in the queue: ",Wq)
    W = Wq + (1 / st)
    print("Average waiting time in the system: ",W)

def main():
    l=float(input("Enter the arrival rate: "))
    m=float(input("Enter the service rate: "))
    print('Enter \n1. For M/M/1 \n2. For M/M/2 \n3. For M/G/1 ')
    k=int(input())
    if(k==1):
        print("\n---M/M/1 Queueing Model---")
        mm1(l,m)
    elif(k==2):
         c=int(input("Enter number of servers (in case of m/m/c): "))
         print("\n---M/M/C Queueing Model---")
         mmc(l,m,c)
    else:
```

```
        print("\n---M/G/1 Queueing Model---")
        mg1(l,m)
main()
```

```
Enter the arrival rate: 4
Enter the service rate: 5
Enter
1. For M/M/1
2. For M/M/2
3. For M/G/1
2
Enter number of servers (in case of m/m/c): 2

---M/M/C Queueing Model---
Probability that the server is busy:  0.4
Probability that the server is idle:  0.6
Expected number of customers in the system:  0.2355555555555556
Expected number of customers in the queue:  0.21432098765432103
Average waiting time in the system:  0.06675062972292192
Average waiting time in the queue:  0.06073327735796251
```

3. Write a Program to implement Inheritance. Create a class Employee inherit two classes Manager and Clerk from Employee.

In [7]:

```
class Employee:
    def __init__(self,ecode,name):
        self.ecode=ecode
        self.name=name
class Manager(Employee):
    def __init__(self,ecode,name,manages):
        Employee.__init__(self,ecode,name)
        self.manages=manages
    def display(self):
        print('Employee number : ',self.ecode)
        print('Employee name : ',self.name)
        print('Manages : ',self.manages)
class Clerk(Employee):
    def __init__(self,ecode,name,works_at):
        Employee.__init__(self,ecode,name)
        self.works_at=works_at
    def display(self):
```

```
        print('Employee number : ',self.ecode)
        print('Employee name : ',self.name)
        print('Works at : ',self.works_at)
obj=Manager(35,'Jeff','Sales')
obj.display()
obj1=Clerk(44,'Aman','Finance')
obj1.display()
```

```
Employee number :  35
Employee name :  Jeff
Manages :  Sales
Employee number :  44
Employee name :  Aman
Works at :  Finance
```

4. Program to fit Poisson distribution on a given data.

In [9]:

```
import scipy.stats as stats

# Define the data
data = [1, 2, 3, 2, 1, 3, 2, 2, 2, 3, 1, 2, 3, 3, 3, 1, 1, 2, 2, 2]

# Fit a Poisson distribution to the data
mu = sum(data)/len(data)  # Estimate the value of mu
poisson_dist = stats.poisson(mu)  # Create the Poisson distribution

# Print the results
print("Estimated value of mu: ", mu)
print("Poisson distribution: ", poisson_dist)
type(poisson_dist)
```

```
Estimated value of mu:  2.05
Poisson distribution:  <scipy.stats._distn_infrastructure.rv_discrete_frozen object at 0x000002411F6112E0>
```

Out[9]:

```
scipy.stats._distn_infrastructure.rv_discrete_frozen
```

In [ ]: