ACTIVITY 1

BWAMBALE BIKALEMA

February 2024

MOUNTAINS OF THE MOON UNIVERSITY
FACULTY OF SCIENCE, TECHNOLOGY AND INNOVATION
DEPARTMENT OF COMPUTER SCIENCE

1 2023/U/MMU/BCS/01674

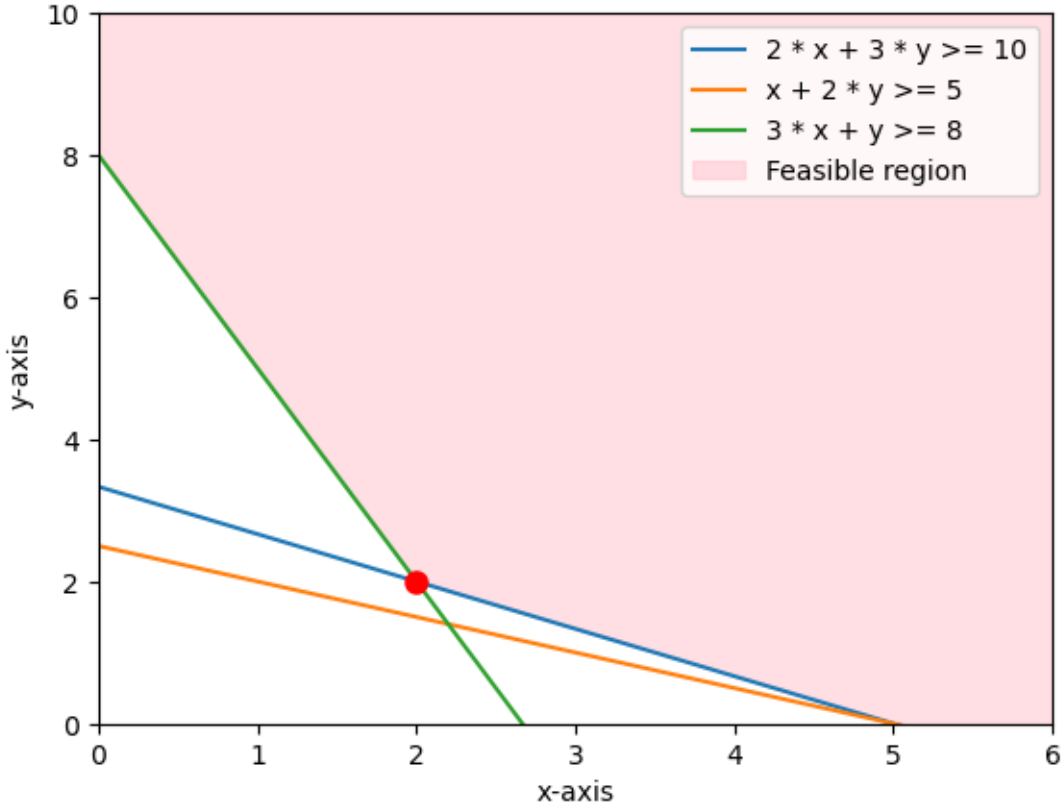
No.1

```
    #importing necessary libraries
from pulp import LpProblem, LpMinimize, LpVariable

#Create the linear programming problem
model = LpProblem(name="Resource_Allocation_Optimization", sense=LpMinimize)

#Define the decision variables
x = LpVariable(name="x") # Resource X
y = LpVariable(name="y") # Resource Y

#Define the objective function
model += 4 * x + 5 * y, "objective"

#Define the constraints
model += 2 * x + 3 * y >= 10,"CPU_constraint"
model += x + 2 * y >= 5,"Memory_constraint"
model += 3 * x + y >= 8,"Storage_constraint"

#Solve the linear programming problem
model.solve()

#Display the Results
print("Optimal solution")
print(f"Resource X (x) :{x.varValue}")
print(f"Resource Y (y) :{y.varValue}")
print(f"Minimum Resource Allocation(Z) :{model.objective.value()}")
```

```
Optimal solution
Resource X (x) :2.0
Resource Y (y) :2.0
Minimum Resource Allocation(Z) :18.0
```

Graph codes

```
# Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

#defining the x array
x = np.linspace(0,16,2000)

#Converting constraints onto inequalities
y1=(10-2*x)/3
y2=(5-x)/2
y3=(8-3*x)

#Plotting contraints
plt.plot(x,y1, label="2 * x + 3 * y >= 10")
plt.plot(x,y2, label="x + 2 * y >= 5")
plt.plot(x,y3, label="3 * x + y >= 8")


#Defining the feasible region
y4 = np.maximum.reduce([y1, y2, y3])
plt.fill_between(x, y4, 10, color="pink", label="Feasible region", alpha=0.5)

#Define limits
plt.xlim(0,6)
plt.ylim(0,10)

#Label and show graph
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```


2 * x + 3 * y >= 10
x + 2 * y >= 5
3 * x + y >= 8
Feasible region
y-axis
x-axis
0
2
4
6
8
10
0
1
2
3
4
5
6

2

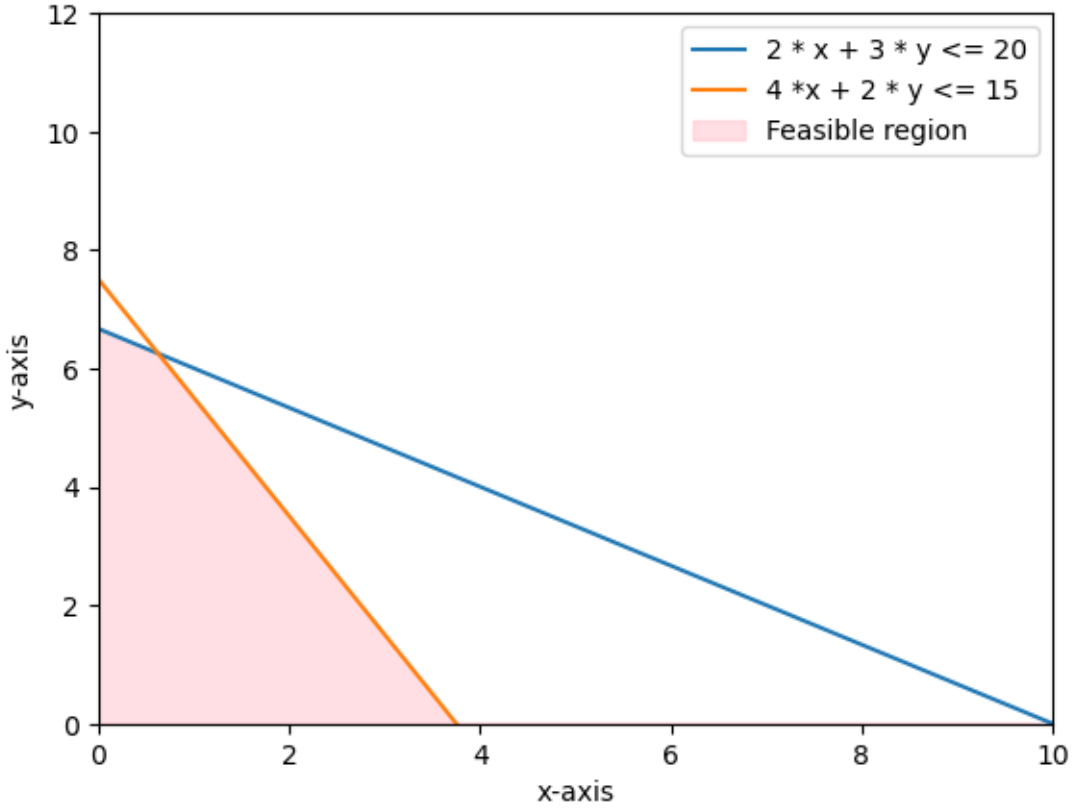
No.2

```
    from pulp import LpProblem, LpMinimize, LpVariable

#Create the linear programming problem
model = LpProblem(name="workloads_Distribution_Optimization", sense=LpMinimize)

#Define decision variables
x = LpVariable(name="x") # Workload X
y = LpVariable(name="y") # Workload Y

#Define the objective function
model += 5 * x + 4 * y, "objective"

#Define the constraints
model += 2 * x + 3 * y <= 20, "Server 1 Capacity constraint"
model += 4 * x + 2 * y <= 15, "Server 2 Capacity constraint"

#Solve the linear programming problem
model.solve()

#Display the results
print("optimum solution")
print(f"workload X(x): {x.varValue}")
print(f"workload Y(y): {y.varValue}")
print(f"minimum workloads distribution(Z): {model.objective.value()}")

optimum solution
workload X(x): 0.0
workload Y(y): 0.0
minimum workloads distribution(Z): 0.0
```

Graph codes

```
    # Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

#defining the x array
x = np.linspace(0,16,2000)

#Converting constraints onto inequalities
y1=(20-2*x)/3
y2=(15-4*x)/2
```

```
#Plotting contraints
plt.plot(x,y1, label="2 * x + 3 * y <= 20")
plt.plot(x,y2, label="4 *x + 2 * y <= 15")

#Defining the feasible region
y3 = np.minimum.reduce([y1, y2,])
plt.fill_between(x, y3, color="pink", label="Feasible region", alpha=0.5)

#Define limits
plt.xlim(0,10)
plt.ylim(0,12)

#Label and show graph
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```


2 * x + 3 * y <= 20
4 *x + 2 * y <= 15
Feasible region
y-axis
x-axis
0
2
4
6
8
10
12

3

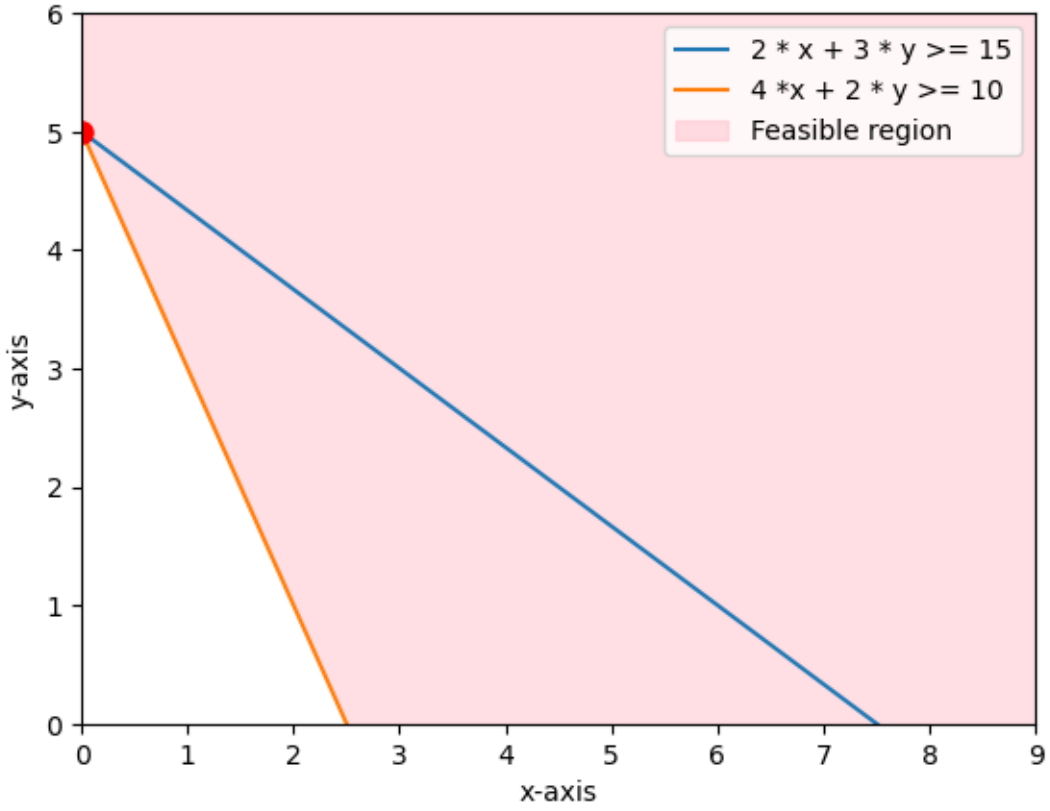
No.3

```
    from pulp import LpProblem, LpMinimize,LpVariable

#create the linear programming problem
model=LpProblem(name="Efficient_resource_allocation_optimization", sense=LpMinimize)

#Define the decition variables
x=LpVariable(name="x") #Resource X
y=LpVariable(name="y") #Resource Y

#Define the objective function
model+=3 * x + 2 * y, "objective"

#Define the constraints
model+= 2 * x + 3 * y>=15, "CPU Allocation constraint"
model+= 4 * x + 2 * y>=10, "Memory allocation constraint"

#Solve the linear programming problem
model.solve()

#Display the results
print("optimal solution")
print(f"Resource X(x) : {x.varValue}")
print(f"Resource Y(y) : {y.varValue}")
print(f"minimum efficient resource allocation(Z): {model.objective.value()}")

optimal solution
Resource X(x) : 0.0
Resource Y(y) : 5.0
minimum efficient resource allocation(Z): 10.0
```

Graph codes

```
   # Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

#defining the x array
x = np.linspace(0,16,2000)

#Converting constraints onto inequalities
y1=(15-2*x)/3
y2=(10-4*x)/2
```

```
#Plotting contraints
plt.plot(x,y1, label="2 * x + 3 * y >= 15")
plt.plot(x,y2, label="4 *x + 2 * y >= 10")

#Defining the feasible region
y3 = np.minimum.reduce([y1, y2,])
plt.fill_between(x, y3, 15, color="pink", label="Feasible region", alpha=0.5)

#Define limits
plt.xlim(0,10)
plt.ylim(0,8)

#Label and show graph
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```


2 * x + 3 * y >= 15
4 *x + 2 * y >= 10
Feasible region
y-axis
x-axis
0
1
2
3
4
5
6
7
8
9

4

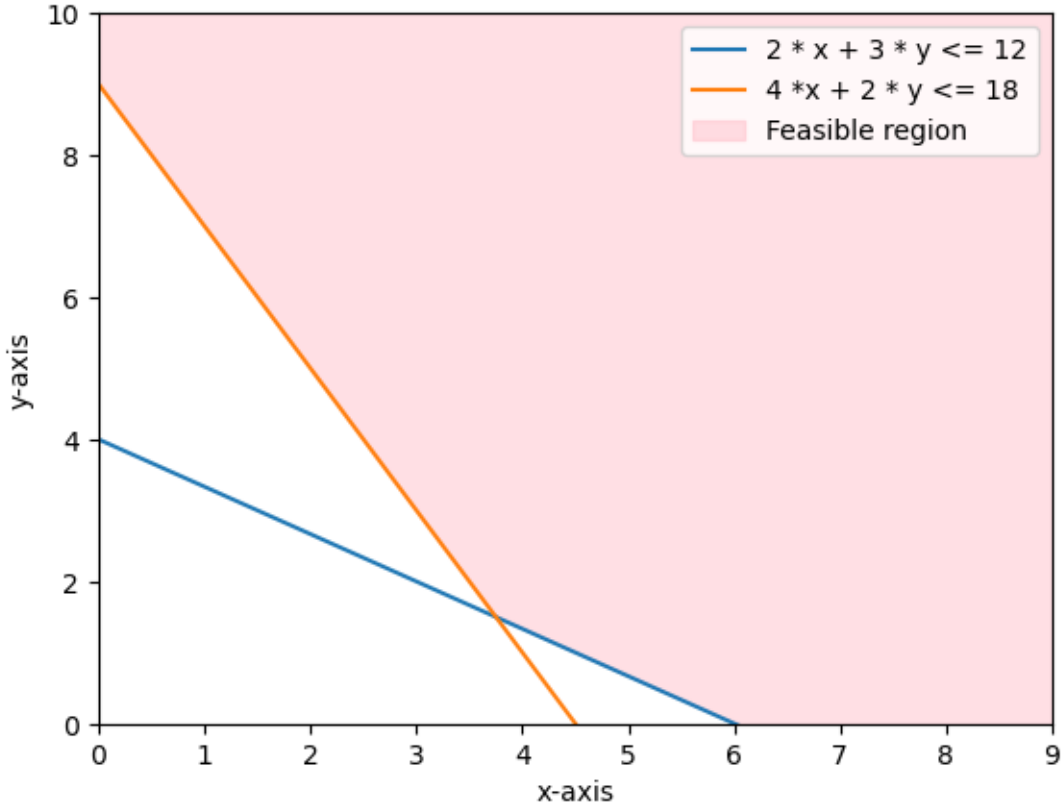
No.4

```
    from pulp import LpProblem, LpMinimize, LpVariable

#Create the linear programming problem
model=LpProblem(name="Tenant_Resouyrce_Sharing_optimization", sense= LpMinimize)

#Define the decision variables
x=LpVariable(name="x") #Resource X
y=LpVariable(name="y") #Resource Y

#Define the objective function
model+= 5 * x + 4 * y, "objective"

#Define the constraints
model+=2 * x + 3 * y >= 12, "Tenant 1 constraint"
model+=4 * x + 2 * y >= 18, "Tenant  constraint"

#Solve the linear programming problem
model.solve()

#Display the results
print("optimal solution")
print(f"Resource X(x) {x.varValue}")
print(f"Resource Y(y) {y.varValue}")
print(f"minimum resource sharing (Z): {model.objective.value()}")

optimal solution
Resource X(x) 3.75
Resource Y(y) 1.5
minimum resource sharing (Z): 24.75
```

Graph codes

```
    # Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

#defining the x array
x = np.linspace(0,16,2000)

#Converting constraints onto inequalities
y1=(12-2*x)/3
y2=(18-4*x)/2
```

```
#Plotting contraints
plt.plot(x,y1, label="2 * x + 3 * y >= 12")
plt.plot(x,y2, label="4 *x + 2 * y >= 18")

#Defining the feasible region
y3 = np.maximum.reduce([y1, y2,])
plt.fill_between(x, y3, 10, color="pink", label="Feasible region", alpha=0.5)

#Define limits
plt.xlim(0,9)
plt.ylim(0,10)

#Label and show graph
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```


2 * x + 3 * y <= 12
4 *x + 2 * y <= 18
Feasible region
y-axis
x-axis
0
1
2
3
4
5
6
7
8
9
10

5

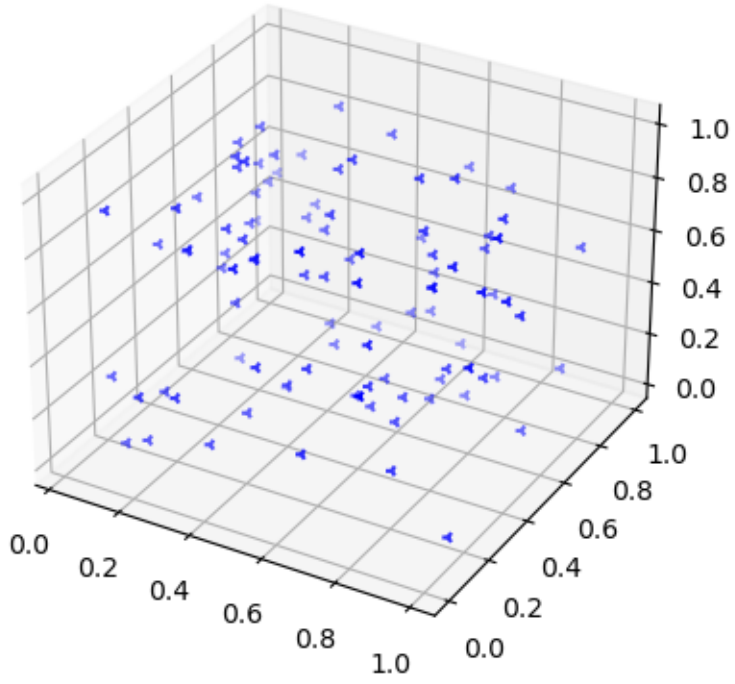
No.5

```
    from pulp import LpProblem, LpMinimize, LpVariable

#Create the linear programming problem
model = LpProblem(name="Production_planning_optimization", sense= LpMinimize)

#Define the decision variables
x1=LpVariable(name="x1", lowBound=0) #Quantity of product 1
x2=LpVariable(name="x2", lowBound=0) #Quantity of product 2
x3=LpVariable(name="x3", lowBound=0) #Quantity of product 3

#Define the objective function
model+= 5 * x1 + 3 * x2 + 4 * x3, "objective"

#Define the constraints
model+= 2 * x1 + 3 * x2 + x3 <=1000, "Raw material constrains"
model+= 4 * x1 + 2 * x2 + 5 * x3 <=120, "Labor hours constrains"
model+= x1>=200,"minimum_x1"
model+= x1>=300,"minimum_x2"
model+= x1>=150,"minimum_x3"

#Solve the linear progrmming problem
model.solve()

#Display the results
print("optimal solution")
print(f"Quantity of product 1(x1) :{x1.varValue}")
print(f"Quantity of product 1(x2) :{x2.varValue}")
print(f"Quantity of product 1(x3) :{x3.varValue}")
print(f"minimum production planning(Z):{model.objective.value()}")

optimal solution
Quantity of product 1(x1) :300.0
Quantity of product 1(x2) :0.0
Quantity of product 1(x3) :0.0
minimum production planning(Z):1500.0
```

Graph codes

```
    import matplotlib.pyplot as plt
import numpy as np

#Generate random data
x1 = np.random.rand(100)
```

```
x2 = np.random.rand(100)
x3 = np.random.rand(100)

fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')

#Craete a 3D scatter plot
ax.scatter(x1,x2,x3, color='blue', marker = '3')

ax.setx1_label('X1_Label')
ax.setx2_label('X2_Label')
ax.setx3_label('X3_Label')

plt.grid()
plt.legend()
plt.show()
```



6

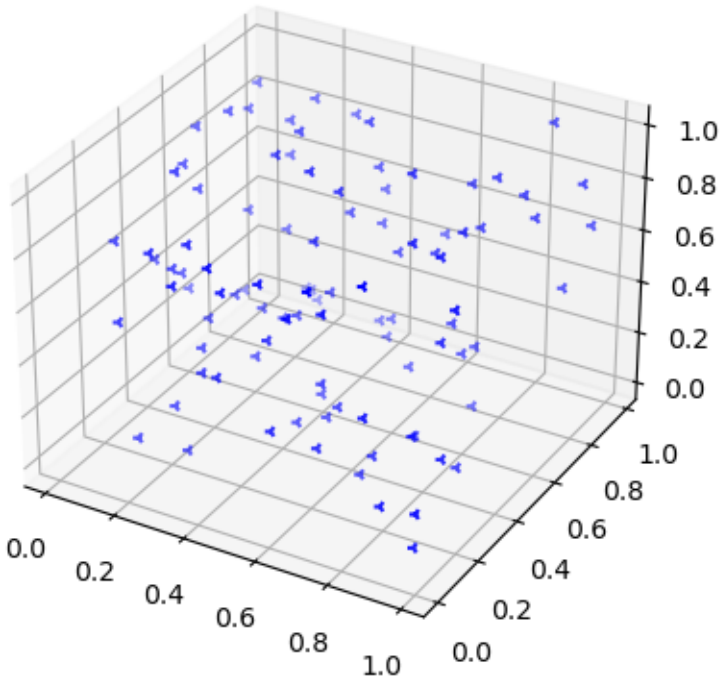
No.6

```
    from pulp import LpProblem,LpMaximize,LpVariable

#Create the linear programming problem
model=LpProblem(name="Financial_Portifolio_Maximization", sense=LpMaximize)

#Define the decision variables
x1=LpVariable(name="x1", lowBound=0) # Investment in stock A
x2=LpVariable(name="x2", lowBound=0) # Investment in stock B
x3=LpVariable(name="x3", lowBound=0) # Investment in stock C

#Define the objective function
model+=0.08 * x1 + 0.1 * x2 + 0.12 * x3, "objective"

#Define the constraints
model+= 2 * x1 + 3 * x2 + x3 <= 10000, "Budget_constraint"
model+= x1>=2000, "minimum_x1"
model+= x2>=1500, "minimum_x2"
model+= x3>=1000, "minimum_x3"

#solve the linear programming problem
model.solve()

#Display the results
print("optimal solution")
print(f"Investment in stock A(x1): {x1.varValue}")
print(f"Investment in stock B(x2): {x2.varValue}")
print(f"Investment in stock C(x3): {x3.varValue}")
print(f"maximum financial portifolio(Z): {model.objective.value()}")

optimal solution
Investment in stock A(x1): 2000.0
Investment in stock B(x2): 1500.0
Investment in stock C(x3): 1500.0
maximum financial portifolio(Z): 490.0
```

Graph codes

```
    import matplotlib.pyplot as plt
import numpy as np

#Generate random data
x1 = np.random.rand(100)
```

```
x2 = np.random.rand(100)
x3 = np.random.rand(100)

fig = plt.figure()
ax = fig.add_subplot(111, projection = '3d')

#Craete a 3D scatter plot
ax.scatter(x1,x2,x3, color='blue', marker = '3')

ax.setx1_label('X1_Label')
ax.setx2_label('X2_Label')
ax.setx3_label('X3_Label')

plt.grid()
plt.legend()
plt.show()
```



7

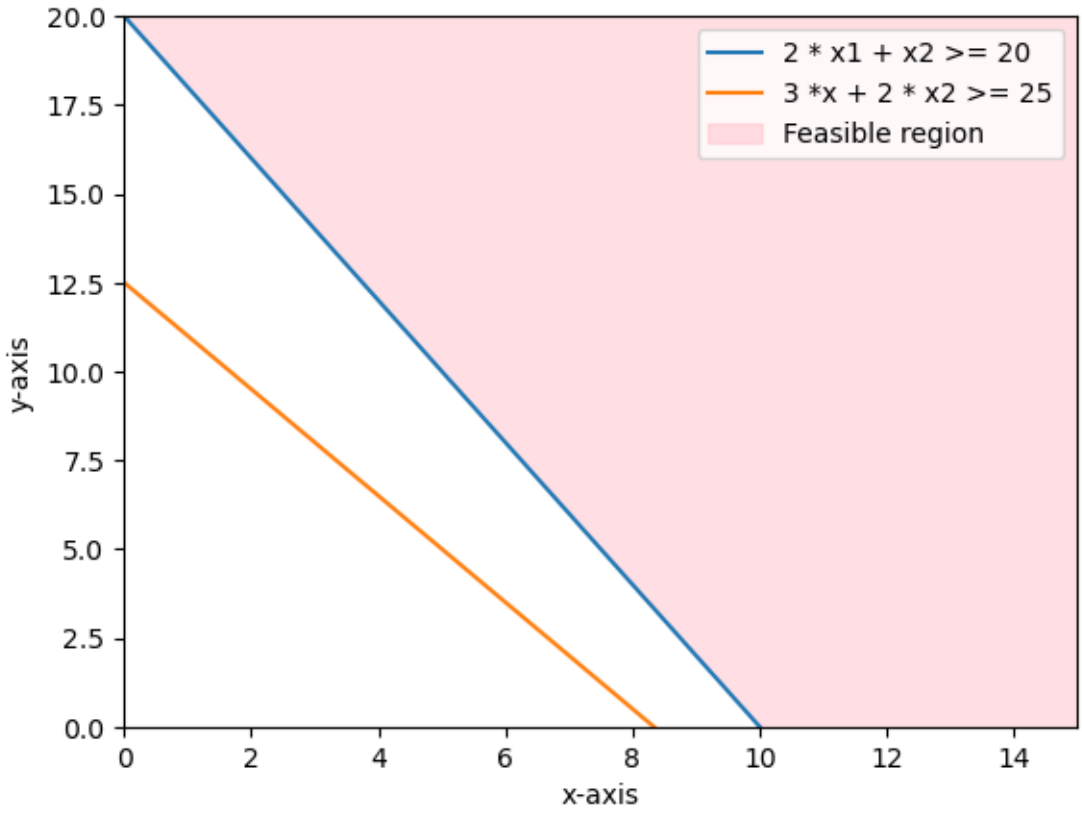
No.7

```
    from pulp import LpProblem, LpMinimize, LpVariable

#Create the linear programming problem
model=LpProblem(name="diet_optimization", sense= LpMinimize)

#Define the decision variable
x1=LpVariable(name="x1", lowBound=0) #Servings of food item 1
x2=LpVariable(name="x2", lowBound=0) #Servings of food item 2

#Define the objective function
model+= 3 * x1 + 2 * x2, "objective"

#Define the constraints
model+= 2 * x1 + x2 >= 20, "protains_constraint"
model+= 3 * x1 + 2 * x2 >= 25, "vitamins _constraint"

#solve the linear programming problem
model.solve()

#Display the results
print("optimal solution")
print(f"servings of food item 1(x1): {x1.varValue}")
print(f"servings of food item 2(x2): {x2.varValue}")
print(f"minimum diet optimization(Z): {model.objective.value()}")

optimal solution
servings of food item 1(x1): 10.0
servings of food item 2(x2): 0.0
minimum diet optimization(Z): 30.0
```

Graph codes

```
    # Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

#defining the x array
x = np.linspace(0,30,2000)

#Converting constraints onto inequalities
x1=(20-2*x)/1
x2=(25-3*x)/2
```

```
#Plotting contraints
plt.plot(x,x1, label="2 * x1 + x2 >= 20")
plt.plot(x,x2, label="3 *x + 2 * x2 >= 25")

#Defining the feasible region
x3 = np.maximum.reduce([x1, x2,])
plt.fill_between(x, x3, 20, color="pink", label="Feasible region", alpha=0.5)

#Define limits
plt.xlim(0,15)
plt.ylim(0,20)

#Label and show graph
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```


2 * x1 + x2 >= 20
3 *x + 2 * x2 >= 25
Feasible region
y-axis
x-axis
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
0
2
4
6
8
10
12
14

8

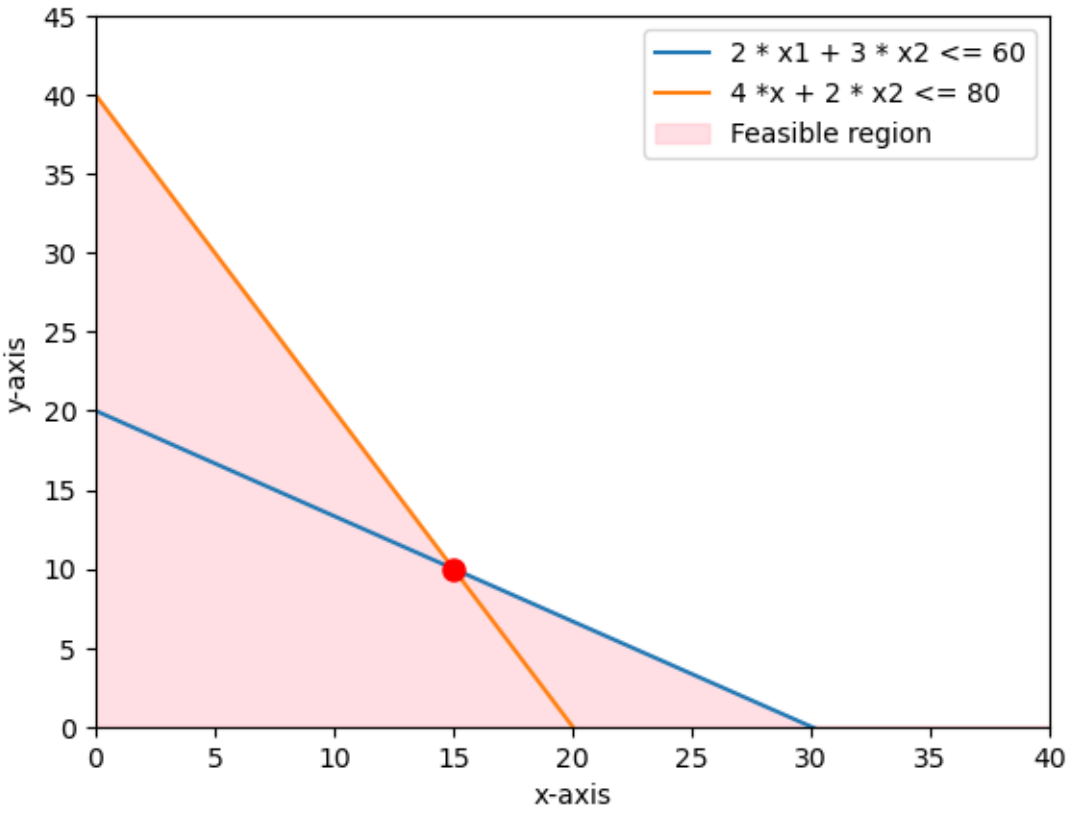
No.8

```
    from pulp import LpProblem, LpMaximize, LpVariable

#Create the linear programming problem
model=LpProblem("Production_planning_optimization", sense=LpMaximize)

#Define the decision variables
x1=LpVariable(name="x1", lowBound=0) #Quantity of product 1
x2=LpVariable(name="x2", lowBound=0) #Quantity of product 2

#Define the objective function
model+= 5 * x1 + 3 * x2, "objective"

#Define the constraints
model+= 2 * x1 + 3 * x2 <= 60, "labor_hours_constraint"
model+= 4 * x1 + 2 * x2 <= 80, "Raw_material_constraint"

#Solve the linear programming problem
model.solve()

#Display results
print("optimal solution")
print(f"Quantity of product 1(x1): {x1.varValue}")
print(f"Quantity of product 2(x2): {x2.varValue}")
print(f"maximum production planning(Z): {model.objective.value()}")

optimal solution
Quantity of product 1(x1): 15.0
Quantity of product 2(x2): 10.0
maximum production planning(Z): 105.0
```

Graph codes

```
    # Importing necessary libraries
import numpy as np
import matplotlib.pyplot as plt

#defining the x array
x = np.linspace(0,100,15000)

#Converting constraints onto inequalities
x1=(60-2*x)/3
x2=(80-4*x)/2
```

```
#Plotting contraints
plt.plot(x,x1, label="2 * x1 + 3 * x2 <= 60")
plt.plot(x,x2, label="4 *x + 2 * x2 <= 80")

#Defining the feasible region
x3 = np.maximum.reduce([x1, x2,])
plt.fill_between(x, x3, 0, color="pink", label="Feasible region", alpha=0.5)

#Define limits
plt.xlim(0,40)
plt.ylim(0,45)

#Label and show graph
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```


2 * x1 + 3 * x2 <= 60
4 *x + 2 * x2 <= 80
Feasible region
y-axis
x-axis
0
5
10
15
20
25
30
35
40
45