MOUNTAINS OF THE MOON UNIVERSITY
FACULTY OF SCIENCE,TECHNOLOGY AND INNOVATION
DEPARTMENT OF COMPUTER SCIENCE
INDIVIDUAL COURSE WORK
REG NO: 2023/U/MMU/BCS/01564
COURSECODE: BCS 1201
COURSEUNIT:LINEAR PROGRAMING
LECTURER: OCEN SAMUEL

Kyambadde Joshua

February 2024

1 ANSWERS

```
 NO.1 BASIC RESOURCE ALLOCATION
#importing necessary libraries
from pulp import LpProblem, LpMinimize, LpVariable

#create a linear programing problem
model = LpProblem(name="Cost_Optimization", sense=LpMinimize)

#define decision variables
x=LpVariable(name="x",lowBound=0)
y=LpVariable(name="y",lowBound=0)

#define objective function
model+= ((4*x)+(5*y)), "objective"

#define constraints
model+= 2*x+3*y>=10, "CPU"
model+= 1*x+2*y>=5, "memory"
model+=3*x+y>=8, "storage"
```

```
#solve the linear programing problem
model.solve()

#display the results
print("Optimal Solution:")
print(f" x (x): {x.varValue}")
print(f" Y (Y): {x.varValue}")
print(f"Minimum(z): {model.objective.value()}")
Optimal Solution:
 x (x): 2.0
 Y (Y): 2.0
Minimum(z): 18.0
\[CODES FOR GRAPH\]
import numpy as np
import matplotlib.pyplot as plt
#convertingconstraintsto inqualities
x=np.linspace(0,15,400)

y1=(20-2*x)/3
y2=(15-4*x)/2

#plot constraints
plt.plot(x,y1,label="2x+3y>=20")
plt.plot(x,y2,label="4x+2y>=15")

#define the feasible region
y3=np.minimum(y1,y2)

plt.fill_between(x,y3,color="blue",alpha=0.5,label="feasible region")

#plot the optimal solution point
optimal_x=2.0
optimal_y=2.0
plt.plot(2.0,2.0,"ro",markersize=8,label="optimal_solution")

#defthe limits
plt.xlim(0,10)
plt.ylim(0,10)
plt.grid(True)
plt.legend()
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("feasible region and optimal solution")
```

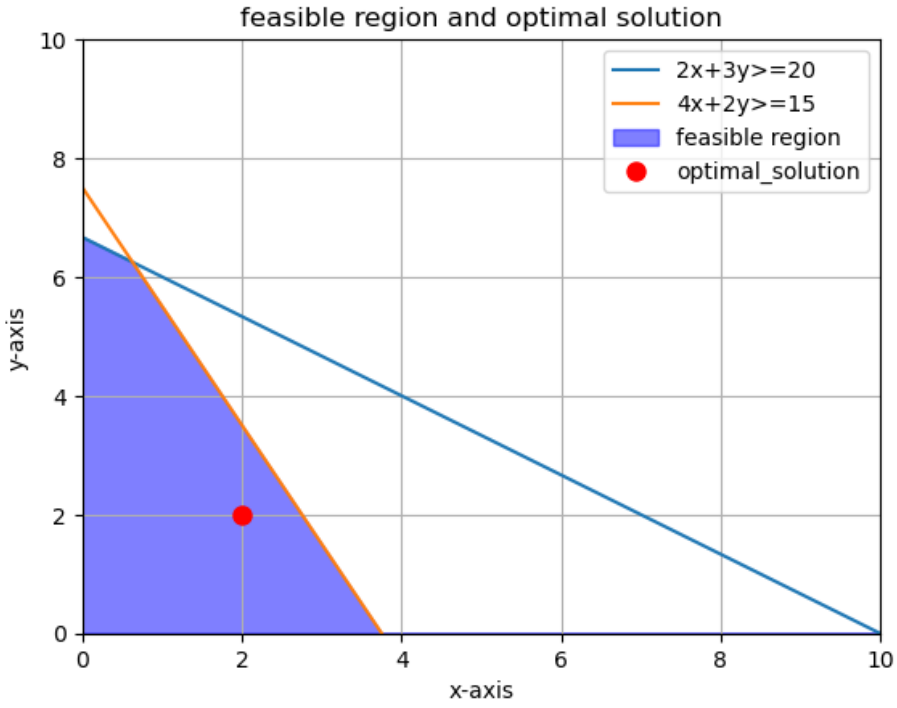

Figure 1: the graph for number one

```
   NO.2
   #importing necessary libraries
from pulp import LpProblem, LpMinimize, LpVariable

#create a linear programing problem
model = LpProblem(name="Time_Optimization", sense=LpMinimize)

#define decision variables
x=LpVariable(name="x",lowBound=0)
y=LpVariable(name="y",lowBound=0)

#define objective function
model+= ((5*x)+(4*y)), "objective"

#define constraints
model+= 2*x+3*y<=20, "server1 capacity"
model+= 4*x+2*y<=15, "server2 capacity"

#solve the linear programing problem
model.solve()

#display the results
print("Optimal Solution:")
print(f"load x (x): {x.varValue}")
print(f"load Y (Y): {x.varValue}")
print(f"Minimum Value(z): {model.objective.value()}")
Optimal Solution:
load x (x): 0.0
load Y (Y): 0.0
Minimum Value(z): 0.0
\[codes for graph\]
import numpy as np
import matplotlib.pyplot as plt
#convertingconstraintsto inqualities
x=np.linspace(0,9,15)

y1=(20-2*x)/3
y2=(15-4*x)/2

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

#define the feasible region
y3=np.minimum(y1,y2)
```

```
plt.fill_between(x,y3,color="gray",alpha=0.5,label="feasible region")

#plot the optimal solution point
optimal_x=0
optimal_y=0
plt.plot(0,0,"ro",markersize=8,label="optimal_solution")

#defthe limits
plt.xlim(0,10)
plt.ylim(0,10)
plt.grid(True)
plt.legend()
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("feasible region and optimal solution")
```

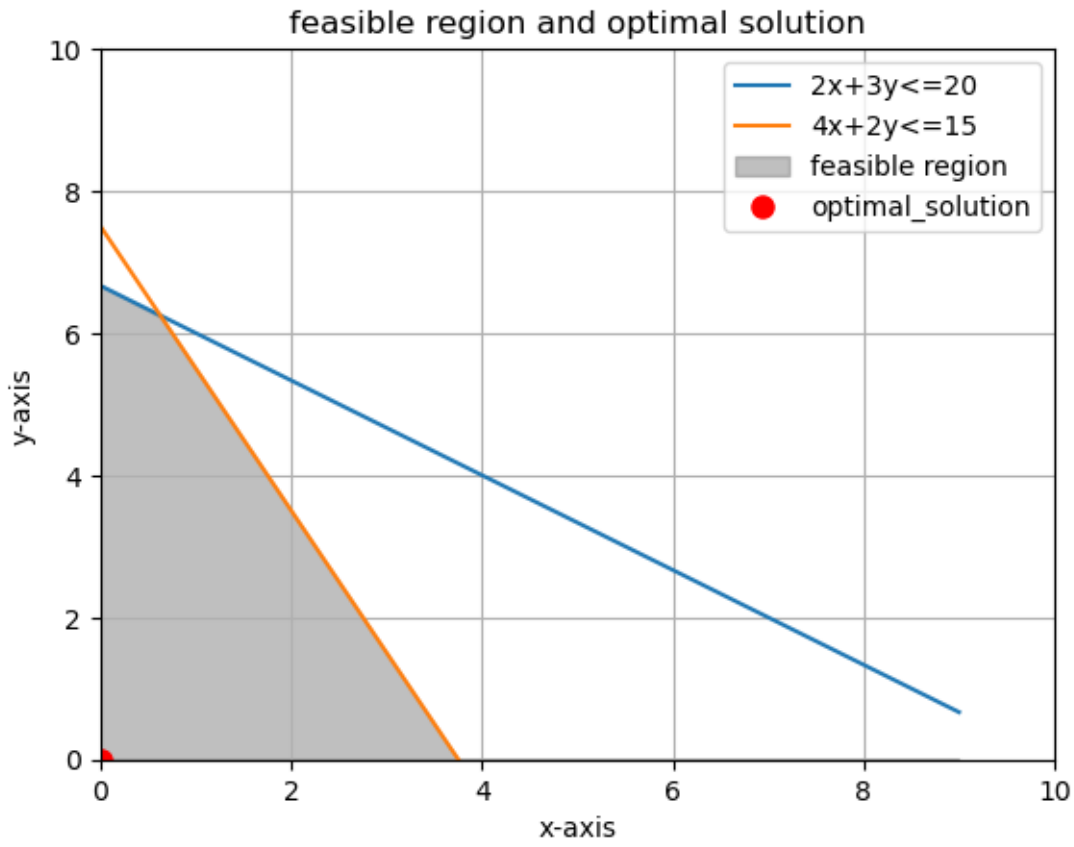

Figure 2: graph for number two

```
    NO.3
   #importing necessary libraries
from pulp import LpProblem, LpMinimize, LpVariable

#create a linear programing problem
model = LpProblem(name="Energy_Optimization", sense=LpMinimize)

#define decision variables
x=LpVariable(name="x",lowBound=0)
y=LpVariable(name="y",lowBound=0)

#define objective function
model+= ((3*x)+(2*y)), "objective"

#define constraints
model+= 2*x+3*y>=15, "CPU Allocation"
model+= 4*x+2*y>=10, "Menory Allocation"

#solve the linear programing problem
model.solve()

#display the results
print("Optimal Solution:")
print(f"x (x): {x.varValue}")
print(f"Y (Y): {x.varValue}")
print(f"Minimum Energy(z): {model.objective.value()}")
Optimal Solution:
x (x): 0.0
Y (Y): 0.0
Minimum Energy(z): 10.0
\[code for the graph\]
import numpy as np
import matplotlib.pyplot as plt
#convertingconstraintsto inqualities
x=np.linspace(0,15,400)

y1=(15-2*x)/3
y2=(10-4*x)/2

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

#define the feasible region
y3=np.minimum(y1,y2)
```

```
plt.fill_between(x,y3,color="gray",alpha=0.5,label="feasible region")

#plot the optimal solution point
optimal_x=0
optimal_y=0
plt.plot(0,0,"ro",markersize=8,label="optimal_solution")

#defthe limits
plt.xlim(0,10)
plt.ylim(0,10)
plt.grid(True)
plt.legend()
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("feasible region and optimal solution")
```

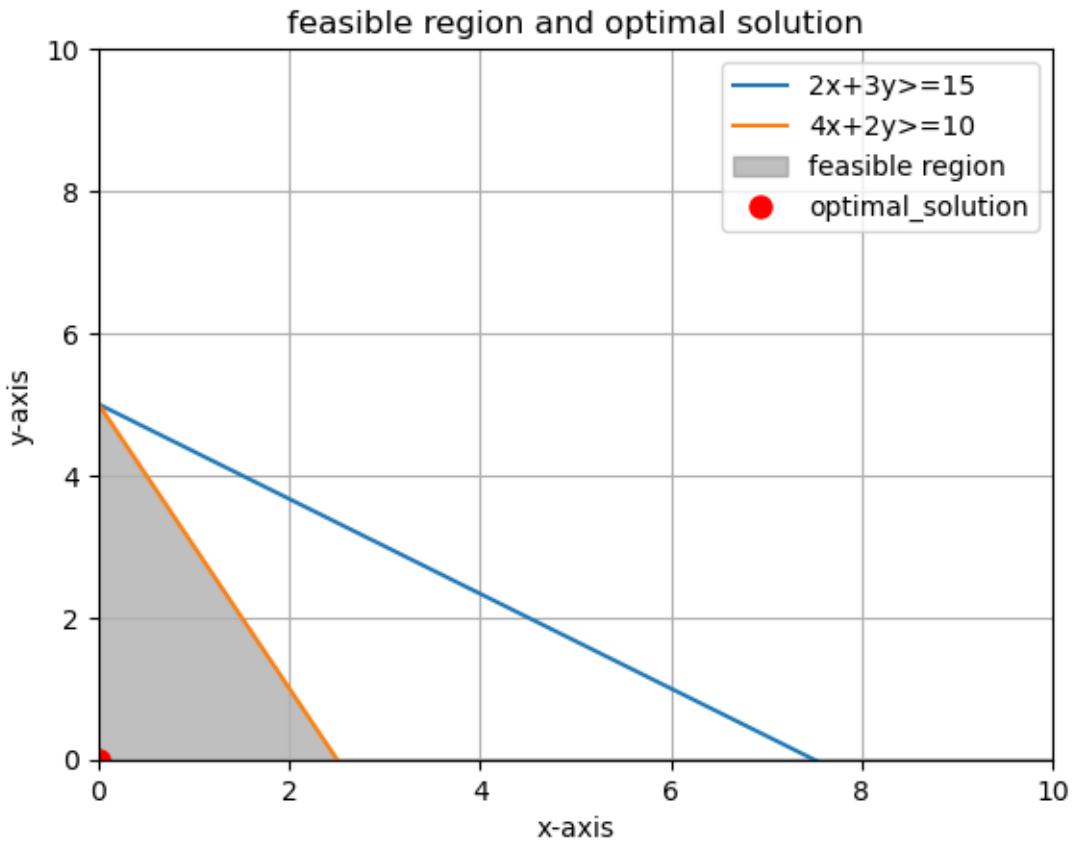

Figure 3: graph for number 3

```
    NO.4
    #importing necessary libraries
from pulp import LpProblem, LpMinimize, LpVariable

#create a linear programing problem
model = LpProblem(name="resource_Optimization", sense=LpMinimize)

#define decision variables
x=LpVariable(name="x",lowBound=0)
y=LpVariable(name="y",lowBound=0)

#define objective function
model+= ((5*x)+(4*y)), "objective"

#define constraints
model+= 2*x+3*y>=12, "tenant 1"
model+= 4*x+2*y>=18, "tenant 2"

#solve the linear programing problem
model.solve()

#display the results
print("Optimal Solution:")
print(f"x (x): {x.varValue}")
print(f"Y (Y): {x.varValue}")
print(f"Minimum (z): {model.objective.value()}")
Optimal Solution:
x (x): 3.75
Y (Y): 3.75
Minimum (z): 24.75
\[CODE FOR THE GRAPH\]
import numpy as np
import matplotlib.pyplot as plt
#convertingconstraintsto inqualities
x=np.linspace(0,15,400)

y1=(12-2*x)/3
y2=(18-4*x)/2

#plot constraints
plt.plot(x,y1,label="2x+3y<=12")
plt.plot(x,y2,label="4x+2y<=18")

#define the feasible region
y3=np.minimum(y1,y2)
```

```
plt.fill_between(x,y3,color="red",alpha=0.5,label="feasible region")

#plot the optimal solution point
optimal_x=3.75
optimal_y=3.75
plt.plot(3.75,3.75,"ro",markersize=8,label="optimal_solution")

#defthe limits
plt.xlim(0,10)
plt.ylim(0,10)
plt.grid(True)
plt.legend()
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("feasible region and optimal solution")
```

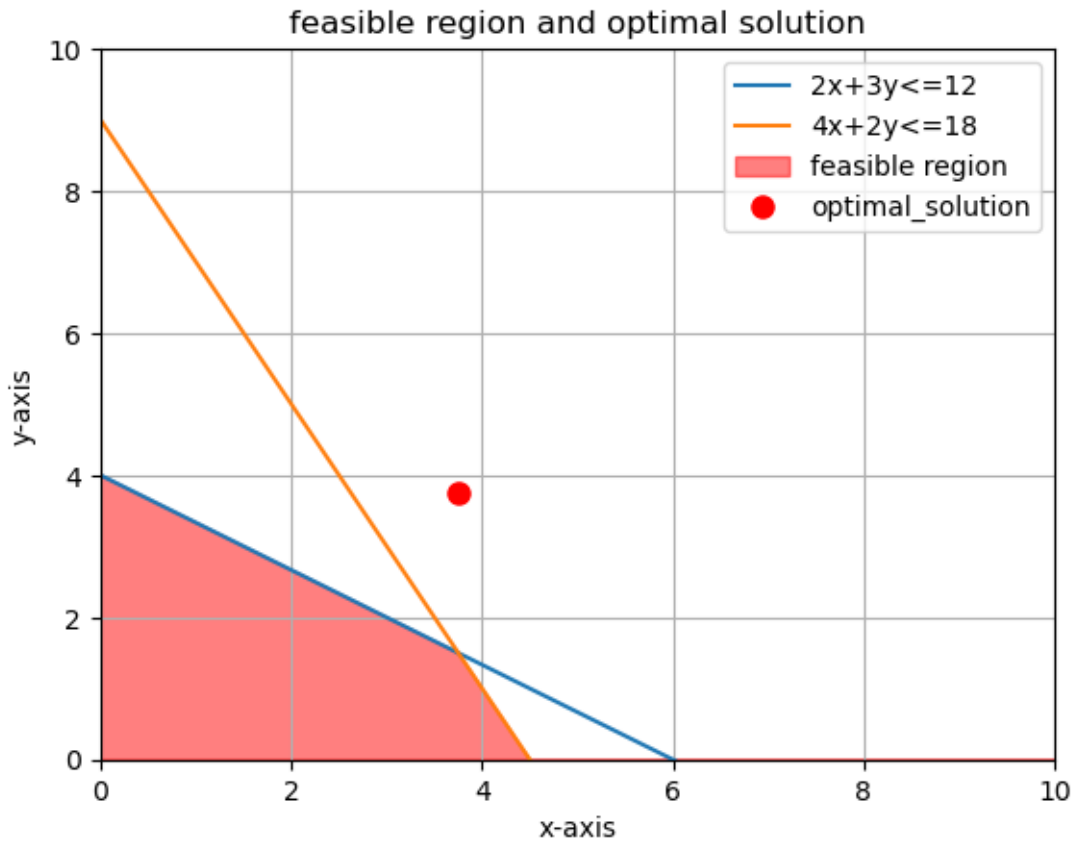

Figure 4: graph for number 4

```
    NO.1 TREANSPORTATION LOGISSTICS
    #importing necessary libraries
from pulp import LpProblem, LpMinimize, LpVariable

#create a linear programing problem
model = LpProblem(name="cost_Optimization", sense=LpMinimize)

#define decision variables
x1=LpVariable(name="x1",lowBound=0)
x2=LpVariable(name="x2",lowBound=0)
x3=LpVariable(name="x3",lowBound=0)
#define objective function
model+= ((5*x1)+(3*x2)+(4*x3)), "objective"

#define constraints
model+= 2*x1+3*x2+1*x3<=1000 # "raw material"
model+= 4*x1+2*x2+5*x3<=120 # "labor hours"
model+= x1 >= 200 #"demand"
model+= x2 >= 300 #"demand"
model+= x3 >= 150 #"demand"

#solve the linear programing problem
model.solve()

#display the results
print("Optimal Solution:")
print(f"x1 (x1): {x1.varValue}")
print(f"x2 (x2): {x2.varValue}")
print(f"x3 (x3): {x3.varValue}")
print(f"Minimum (z): {model.objective.value()}")
Optimal Solution:
x1 (x1): 200.0
x2 (x2): 300.0
x3 (x3): 0.0
Minimum (z): 1900.0
\[CODE FOR THE GRAPH\]
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

#define constraints
def constraint1(x1,x2):
    return (1000-2*x1-3*x2)/1

def constraint2(x1,x2):
    return (120-4*x1-2*x2)/5
```

```
def constraint3(x):
    return np.full_like(x, 200)

def constraint4(x):
    return np.full_like(x, 300)

def constraint5(x):
    return np.full_like(x, 150)

#define objective function
def objective_function(x1, x2):
    return 5*x1+3*x2+4*(1000-2*x1-3*x2)/5

#genarate x, y values for ploting
x_values = np.linspace(0,500,100)
y_values = np.linspace(0,500,100)
x,y = np.meshgrid(x_values,y_values)

#plot the constraints
fig = plt.figure()
ax = fig.add_subplot(111,projection='3d')

ax.plot_surface(x,y, constraint1(x,y), alpha=0.5, rstride=100,cstride=100)
ax.plot_surface(x,y, constraint2(x,y), alpha=0.5, rstride=100,cstride=100)
ax.plot_surface(constraint3(x_values),y,x,  alpha=0.5, rstride=100,cstride=100)

ax.plot_surface(x,constraint4(y_values),x,  alpha=0.5, rstride=100,cstride=100)
ax.plot_surface(x,y, constraint5(x), alpha=0.5, rstride=100,cstride=100)

#plot the feasible region
ax.plot_surface(x,y,np.minimum(constraint1(x,y),constraint2(x,y)),color='red',label='Optimum

#Add labels and legend
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title('Feasible Region and Optimum Point')


#show plot
plt.show()
```

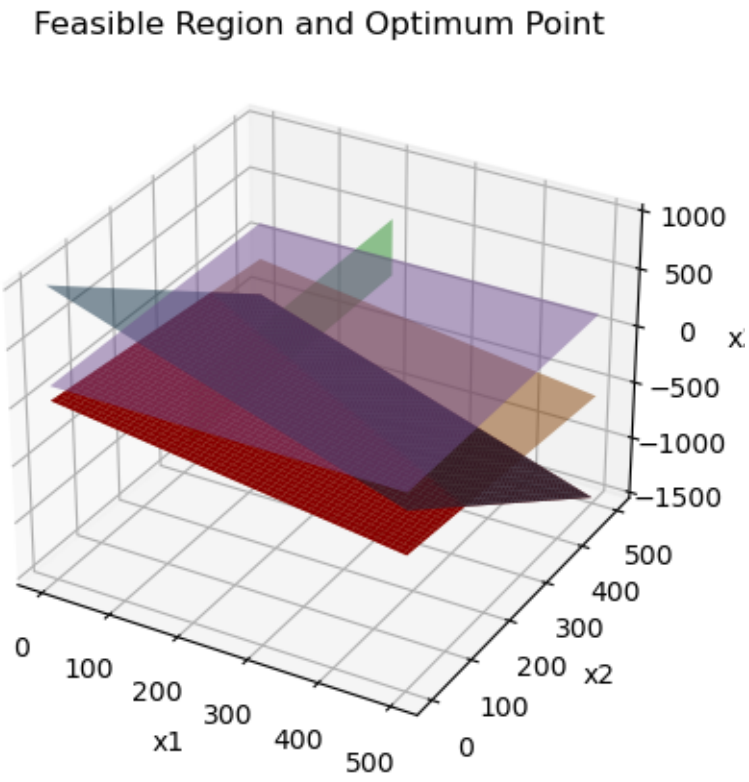

Figure 5: graph 3D

```
    no.2
    #importing necessary libraries
from pulp import LpProblem, LpMaximize, LpVariable

#create a linear programing problem
model = LpProblem(name="investment_Optimization", sense=LpMaximize)

#define decision variables
x1=LpVariable(name="x1",lowBound=0)
x2=LpVariable(name="x2",lowBound=0)
x3=LpVariable(name="x3",lowBound=0)
#define objective function
model+= ((0.08*x1)+(0.1*x2)+(0.12*x3)), "objective"

#define constraints
model+= 2*x1+3*x2+1*x3<=10000 # "maximum budget for investments"
model+= x1 >= 2000#"minimum investment"
model+= x2 >= 1500#"minimum investment"
model+= x3 >= 1000#"minimum investment"

#solve the linear programing problem
model.solve()

#display the results
print("Optimal Solution:")
print(f"x1 (x1): {x1.varValue}")
print(f"x2 (x2): {x2.varValue}")
print(f"x3 (x3): {x3.varValue}")
print(f"Minimum (z): {model.objective.value()}")
x1 (x1): 2000.0
x2 (x2): 1500.0
x3 (x3): 1500.0
Minimum (z): 490.0
\[code for the graph\]
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Define the constraints
x1 = np.linspace(0, 5000, 100)  # adjust the range according to your constraint
x2 = np.linspace(0, 5000, 100)  # adjust the range according to your constraint
x1, x2 = np.meshgrid(x1, x2)

# Define the constraints
constraint1 = (10000 - 2*x1 - 3*x2) / 1
constraint2 = 2000 / 1
```

```
constraint3 = 1500 / 1
constraint4 = 1000 / 1

# Define the objective function
objective_function = 0.08*x1 + 0.1*x2 + 0.12*(10000 - 2*x1 - 3*x2)

# Plot the constraints
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(x1, x2, constraint1, alpha=0.5)
ax.plot_surface(x1, x2, constraint2, alpha=0.5)
ax.plot_surface(x1, x2, constraint3, alpha=0.5)
ax.plot_surface(x1, x2, constraint4, alpha=0.5)

# Plot the objective function
ax.plot_surface(x1, x2, objective_function, alpha=0.5)

ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('Value')

plt.show()
```

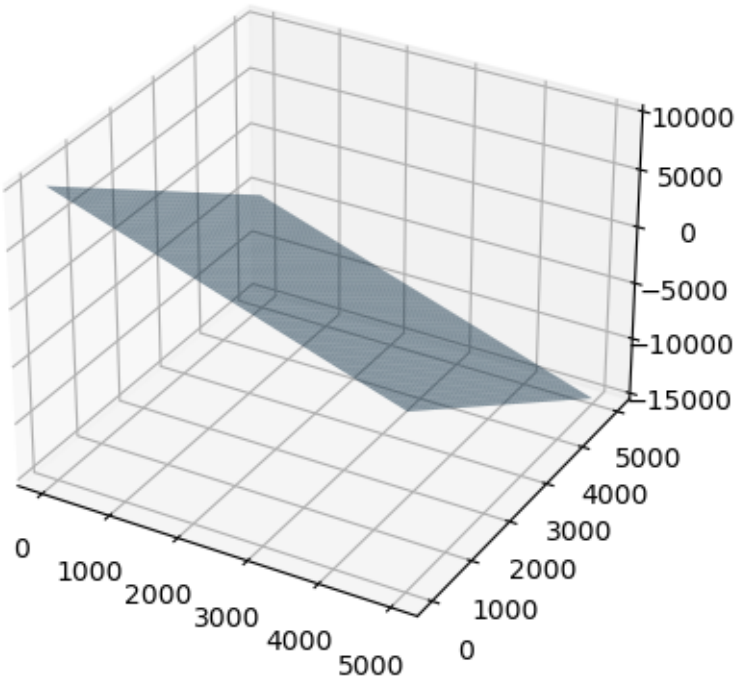

Figure 6: graph

```
    no.1
    project resource allocation
    #importing necessary libraries
from pulp import LpProblem, LpMinimize, LpVariable

#create a linear programing problem
model = LpProblem(name="diet_Optimization", sense=LpMinimize)

#define decision variables
x1=LpVariable(name="x1",lowBound=0)
x2=LpVariable(name="x2",lowBound=0)

#define objective function
model+= ((3*x1)+(2*x2)), "objective"

#define constraints
model+= 2*x1+1*x2>=20, "proteins"
model+= 3*x1+2*x2>=25, "vitamin"

#solve the linear programing problem
model.solve()

#display the results
print("Optimal Solution:")
print(f"x1 (x1): {x.varValue}")
print(f"x2 (x2): {x2.varValue}")
print(f"Minimum (z): {model.objective.value()}")
Optimal Solution:
x1 (x1): 10.0
x2 (x2): 0.0
Minimum (z): 30.0
\[code for the graph\]
import numpy as np
import matplotlib.pyplot as plt
#convertingconstraintsto inqualities
x=np.linspace(0,15,400)

y1=(20-2*x)/1
y2=(25-3*x)/2

#plot constraints
plt.plot(x,y1,label="2x+y<=20")
plt.plot(x,y2,label="3x+2y<=25")

#define the feasible region
y3=np.minimum(y1,y2)
```

```
plt.fill_between(x,y3,color="grey",alpha=0.5,label="feasible region")

#plot the optimal solution point
optimal_x=10.0
optimal_y=0.0
plt.plot(10.0,0.0,"ro",markersize=8,label="optimal_solution")

#defthe limits
plt.xlim(0,15)
plt.ylim(0,15)
plt.grid(True)
plt.legend()
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("feasible region and optimal solution")
```

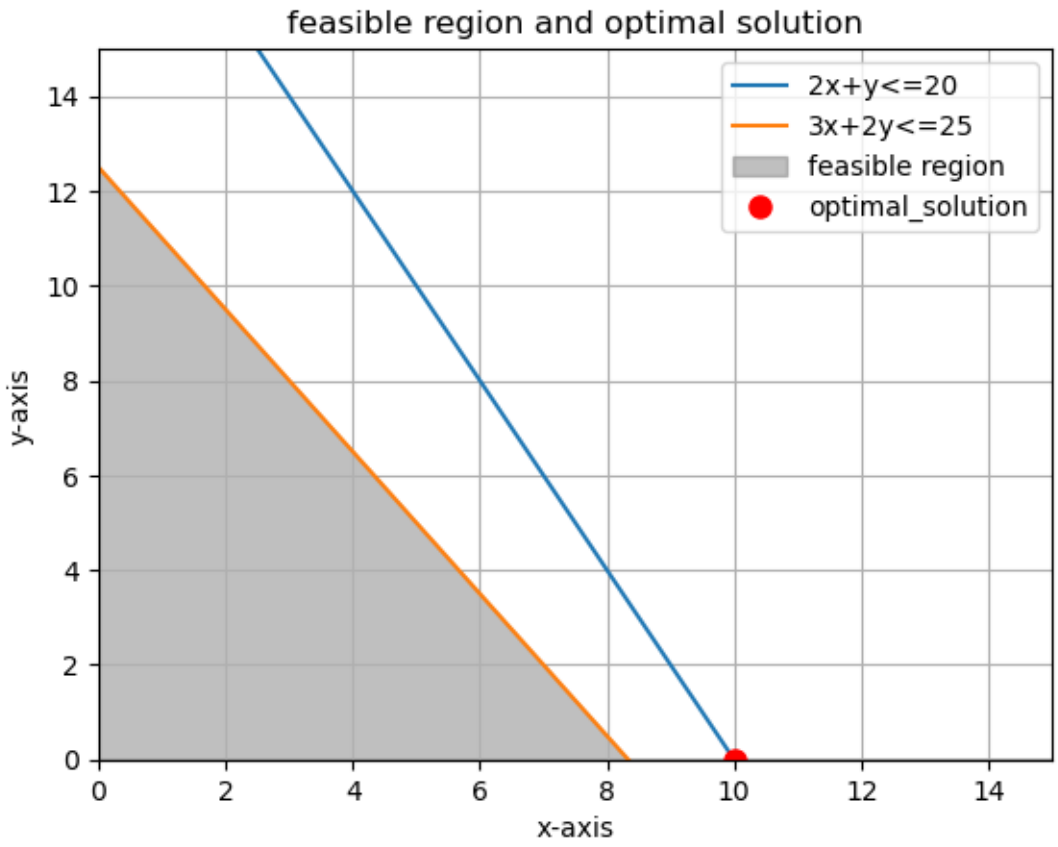

Figure 7: graph

```
    NO.2
  #importing necessary libraries
from pulp import LpProblem, LpMaximize, LpVariable

#create a linear programing problem
model = LpProblem(name="profit_Optimization", sense=LpMaximize)

#define decision variables
x1=LpVariable(name="x1",lowBound=0)
x2=LpVariable(name="x2",lowBound=0)

#define objective function
model+= ((5*x1)+(3*x2)), "objective"

#define constraints
model+= 2*x1+3*x2<=60, "labor"
model+= 4*x1+2*x2<=80, "raw materials"

#solve the linear programing problem
model.solve()

#display the results
print("Optimal Solution:")
print(f"x1 (x1): {x1.varValue}")
print(f"x2 (x2): {x2.varValue}")
print(f"Minimum (z): {model.objective.value()}")
Optimal Solution:
x1 (x1): 15.0
x2 (x2): 10.0
Minimum (z): 105.0
\[CODE FOR THE GRAPH\]
import numpy as np
import matplotlib.pyplot as plt
#convertingconstraintsto inqualities
x=np.linspace(0,15,400)

y1=(60-2*x)/3
y2=(80-4*x)/2

#plot constraints
plt.plot(x,y1,label="2x1+3X2<=60")
plt.plot(x,y2,label="4x1+2X2<=80")

#define the feasible region
y3=np.minimum(y1,y2)
```

```
plt.fill_between(x,y3,color="grey",alpha=0.5,label="feasible region")

#plot the optimal solution point
optimal_x=15.0
optimal_y=10.0
plt.plot(15.0,10.0,"ro",markersize=8,label="optimal_solution")

#defthe limits
plt.xlim(0,15)
plt.ylim(0,19)
plt.grid(True)
plt.legend()
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.title("feasible region and optimal solution")
```

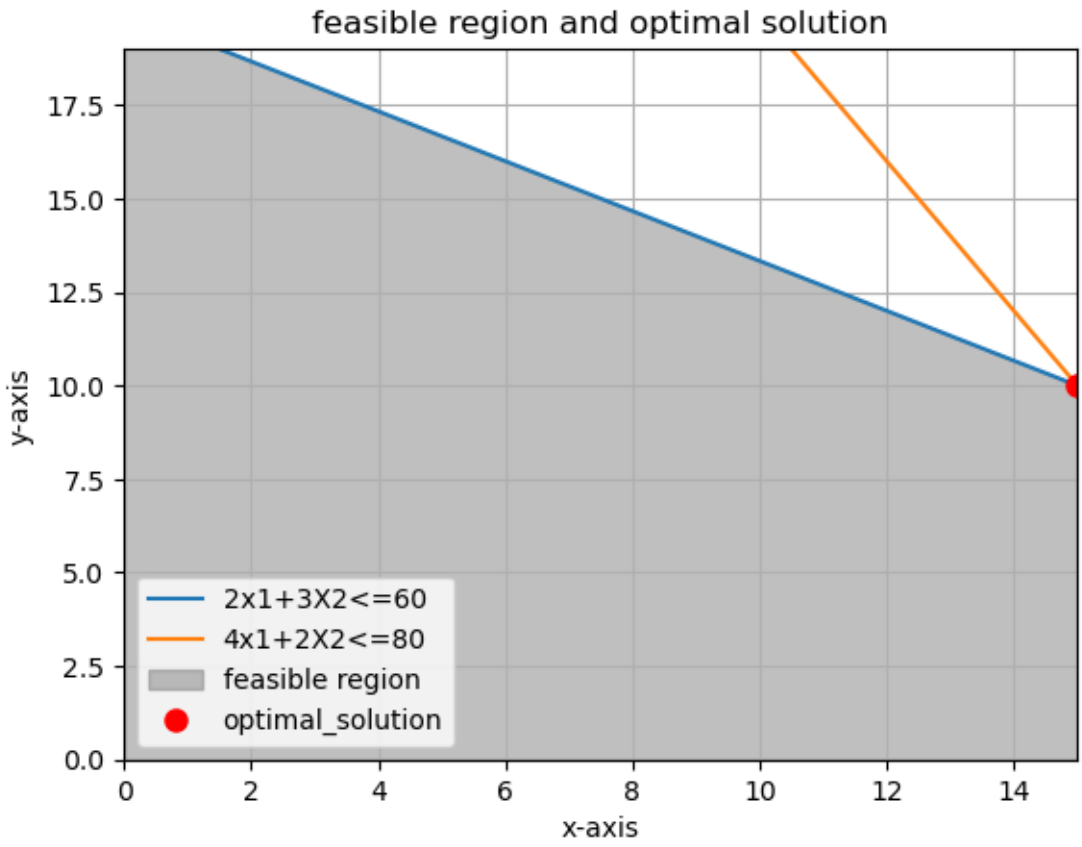

Figure 8: GRAPH