MOUNTAINS OF THE MOON UNIVERSITY
FACULTY OF SCIENCE,TECHNOLOGY AND INNOVATION
DEPARTMENT OF COMPUTER SCIENCE
INDIVIDUAL COURSEWORK
REG NO: 2023/U/MMU/BCS/01661
COURSECODE: BCS 1201
COURSEUNIT:LINEAR PROGRAMMING
LECTURER'S NAME: MR.OCEN SAMUEL

NAME: NAMUGALU CAROLINE

February 2024

1 ANSWERS

```
 NO.1 BASIC RESOURCE ALLOCATION

 from pulp import LpProblem,LpMinimize,LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="basic_resource_allocation",sense=LpMinimize)
#variables
x=LpVariable("x",0)
y=LpVariable("y",0)
#objective function
model+= 4*x + 5*y
#Constraints
model+= 2*x + 3*y >= 10,"CPU Constraint"
model+= x + 2*y >= 5,"Memory Constraint"
model+= 3*x + y >=8,"Storage Constraint"
#Solving
model.solve()
```

```
#resluts
optimal_x=x.varValue
optimal_y=y.varValue
optimal_value = model.objective.value()
print("optimal solution:")
print("x:",optimal_x)
print("y:",optimal_y)
print("z:",optimal_value)
optimal solution:
x: 2.0
y: 2.0
z: 18.0

\[CODES FOR GRAPH\]
import matplotlib.pyplot as plt
import numpy as np
#define x array
x=np.linspace(0,16,20000)
#convert constraints to inequalities
y1=(10-2*x)/3
y2=(5-x)/2
y3=(8-3*x)
#plot constraints
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")
# define the feasible region
y4 =np.maximum.reduce([y1,y2,y3])
plt.fill_between(x,y4,11,color="brown",label="feasible region",alpha=0.3)
# define limits
plt.xlim(0,11)
plt.ylim(0,11)
plt.grid(True)
plt.xlabel ("x-axis")
plt.ylabel ("y-axis")
plt.legend()
plt.show()
```

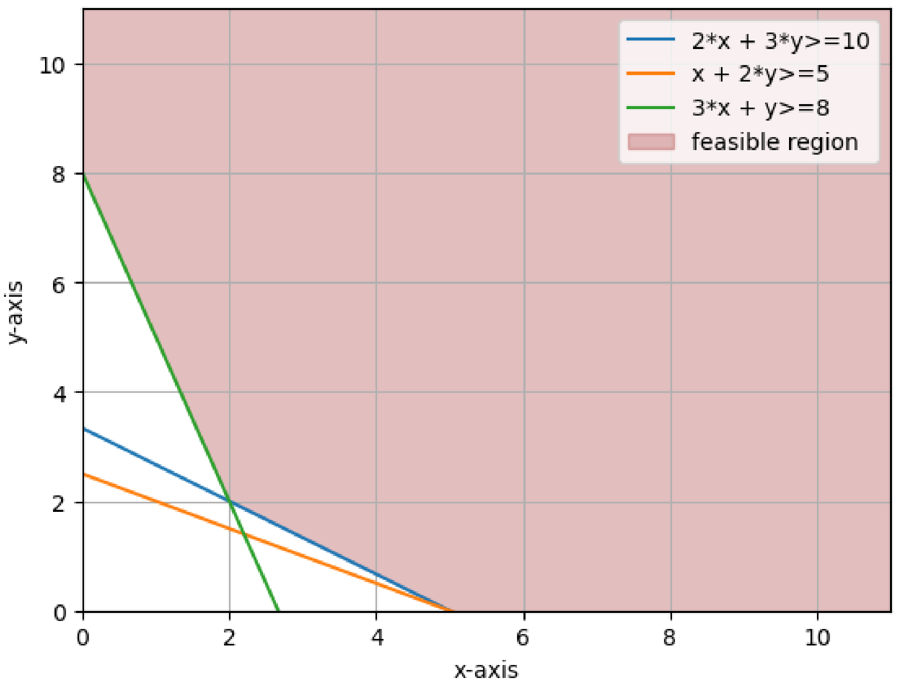

Figure 1: Graph of number one

NO.2 LOAD BALANCING

```
    from pulp import LpProblem,LpMinimize,LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="Loading_balancing",sense=LpMinimize)
#variables
x=LpVariable("x",0)
y=LpVariable("y",0)
#objective function
model+= 5*x + 4*y
#Constraints
model+= 2*x + 3*y <= 20,"Server1 Capacity Constraint"
model+= 4*x + 2*y <= 15," Server2 CapacityConstraint"

#Solving
model.solve()
#resluts
optimal_x=x.varValue
optimal_y=y.varValue
optimal_value = model.objective.value()
print("optimal solution:")
print("x:",optimal_x)
print("y:",optimal_y)
print("z:",optimal_value)
optimal solution:
x: 0.0
y: 0.0
z: 0.0

\[CODES FOR GRAPH\]
import matplotlib.pyplot as plt
import numpy as np
#define x array
x=np.linspace(0,15,10000)
#convert constraints to inequalities
y1=(20-2*x)/3
y2=(15-4*x)/2
#plot constraints
plt.plot(x,y1,label= "2*x +3*y <=20")
plt.plot(x,y2,label= "4*x + 2*y<=15")
# define the feasible region
y3 =np.minimum.reduce([y1,y2])
plt.fill_between(x,y3,color="red",label="feasible region",alpha=0.3)
# define limits
```

```
plt.xlim(0,12)
plt.ylim(0,12)
plt.grid(True)
plt.xlabel ("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```

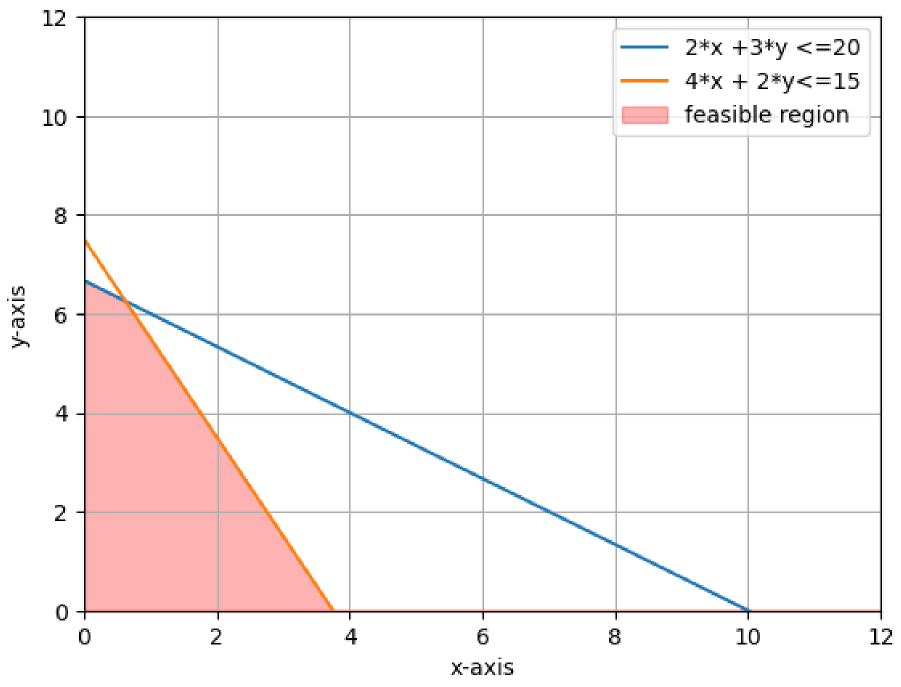

Figure 2: Graph for number two

NO.3 ENERGY EFFICIENT RESOURCE ALLOCATION

```
    from pulp import LpProblem,LpMinimize,LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="Energy-Efficient_resource_allocation",sense=LpMinimize)
#variables
x=LpVariable("x",0)
y=LpVariable("y",0)
#objective function
model+= 3*x + 2*y
#Constraints
model+= 2*x + 3*y >= 15,"CPU Allocation Constraint"
model+= 4*x + 2*y >= 10," Memory Allocation Constraint"
#Solving
model.solve()
#resluts
optimal_x=x.varValue
optimal_y=y.varValue
optimal_value = model.objective.value()
print("optimal solution:")
print("x:",optimal_x)
print("y:",optimal_y)
print("z:",optimal_value)
optimal solution:
x: 0.0
y: 5.0
z: 10.0
```

\[CODES FOR GRAPH\]

```
import matplotlib.pyplot as plt
import numpy as np
#define x array
x=np.linspace(0,17,15000)
#convert constraints to inequalities
y1=(15-2*x)/3
y2=(10-4*x)/2
#plot constraints
plt.plot(x,y1,label="2*x + 3*y >=15")
plt.plot(x,y2,label="4*x + 2*y >=10")
#define the feasible region
y3 =np.maximum.reduce([y1,y2])
plt.fill_between(x,y3,15,color="purple",label="feasible region",alpha=0.3)
# define limits
plt.xlim(0,15)
plt.ylim(0,15)
plt.grid(True)
plt.xlabel ("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```

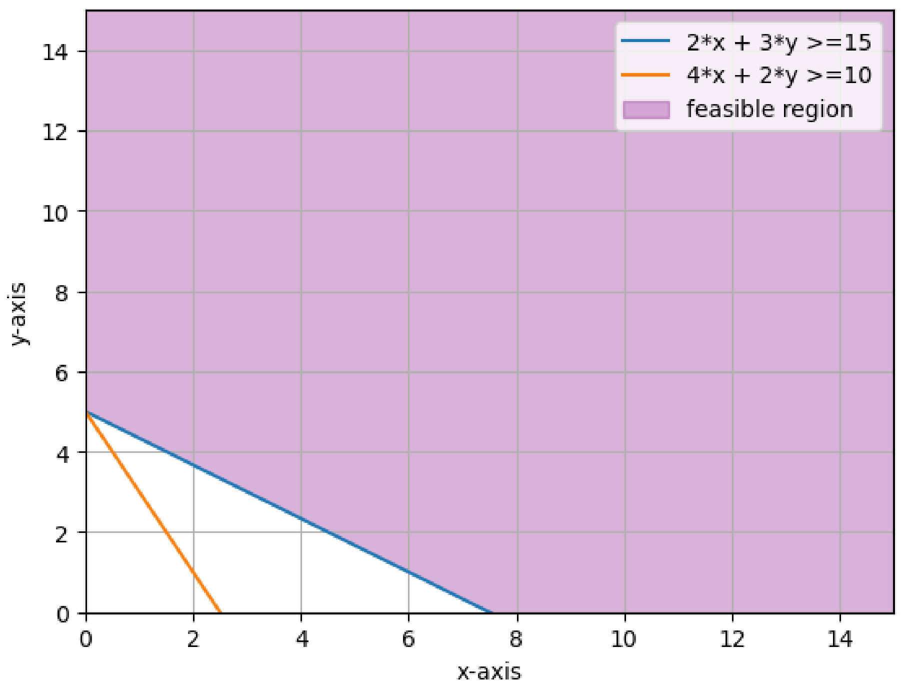

Figure 3: graph for number three

```
NO 4.MULTI-TENANT RESOURCE SHARING
from pulp import LpProblem,LpMinimize,LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model=LpProblem(name="Multi_Tenant_Resource_Sharing",sense=LpMinimize)
#variables
x=LpVariable("x",0)
y=LpVariable("y",0)
#objective function
model+= 5*x + 4*y
#Constraints
model+= 2*x + 3*y >= 12,"Tenant 1 Constraint"
model+= 4*x + 2*y >= 18,"Tenant 2 Constraint"
#Solving
model.solve()
#resluts
optimal_x=x.varValue
optimal_y=y.varValue
optimal_value = model.objective.value()
print("optimal solution:")
print("x:",optimal_x)
print("y:",optimal_y)
print("z:",optimal_value)
optimal solution:
x: 3.75
y: 1.5
z: 24.75

\[CODES FOR GRAPH\]
import matplotlib.pyplot as plt
import numpy as np
#define the x array
x= np.linspace(0,15,20000)
#convert constraints to inequalities
y1=(12-2*x)/3
y2=(18-4*x)/2
#=plot constraints
plt.plot(x,y1,label="2*x + 3*y>=12")
plt.plot(x,y2,label="4*x + 2*y>=18")
#define the feasible region
y3 =np.maximum.reduce([y1,y2])
plt.fill_between(x,y3,13,color="yellow",label="feasible region",alpha=0.3)
#define limits
plt.xlim(0,15)
plt.ylim(0,12)
plt.grid(True)
plt.xlabel ("x-axis")
plt.ylabel ("y-axis")
plt.legend()
plt.show()
```

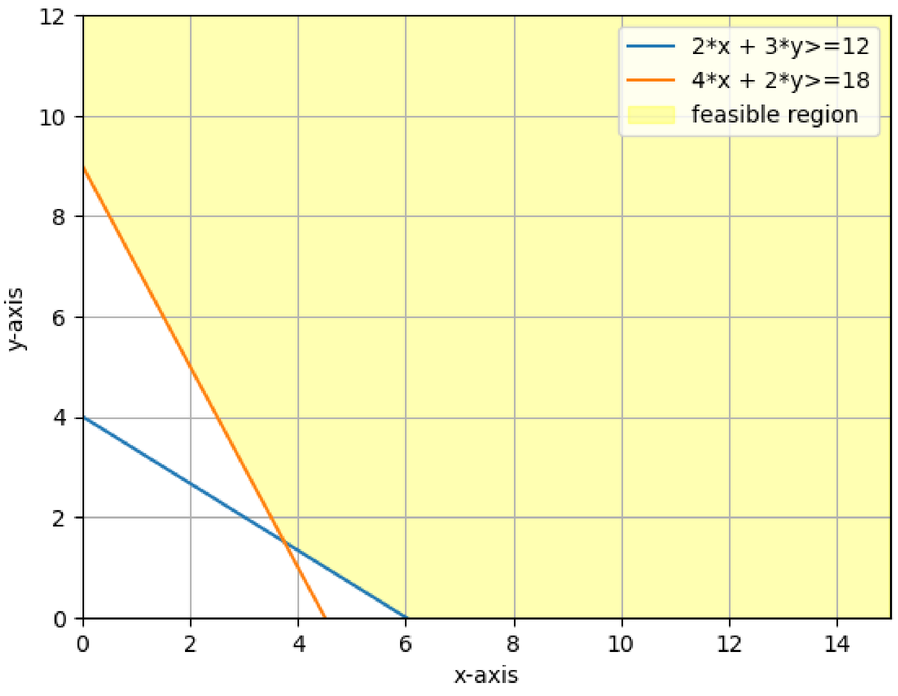

Figure 4: Graph for number four

```
NO. 5 PRODUCTION PLANNING IN MANUFACTURING
from pulp import LpVariable,LpMinimize,LpProblem
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model=LpProblem("Production_Planning_in_Manufacturing",sense= LpMinimize)
#LpVariables
x1=LpVariable("x1",0)
x2=LpVariable("x2",0)
x3=LpVariable("x3",0)
#Objective function
model+=5*x1 + 3*x2 + 4*x3
#Constraints
model+=2*x1 + 3*x2 + x3 <=1000,"Raw_material_constraint"
model+=4*x1 + 2*x2 + 5*x3 <=120,"labour_hours_constraint"
model+=x1>=200,"minimum_x1"
model+=x2>=300,"minimum_x2"
model+=x3>=150,"minimum_x3"
#solving
model.solve()
#results
optimal_x1=x1.varValue
optimal_x2=x2.varValue
optimal_x3=x3.varValue
optimal_value=model.objective.value()
print("optimal solution")
print("x1",optimal_x1)
print("x2",optimal_x2)
print("x3",optimal_x3)
print("z:",optimal_value)
optimal solution
x1 200.0
x2 300.0
x3 0.0
z: 1900.0


\[codes for graph\]
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

#Define the constraints
def constraint1(x1):
    return (1000 - 2* x1)/3
```

```
def constraint2(x1):
    return (25 - 4* x1)/5

def constraint3(x1):
    return 300

def constraint4(x1):
    return 150

#create arrays of x1 values for plotting
x1_values = np.linspace(0, 400, 1000)

#calculate corresponding x2 and x3 values based on the constraints
x2_values = constraint1(x1_values)
x3_values = constraint2(x1_values)

#plot the constraints
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111,projection='3d')

#plot the feasible region surface
ax.plot_surface(x1_values.reshape(-1,1), np.full_like(x1_values,300), x3_values.reshape(-1,

#Define the unwanted region
x1_unwanted = np.linspace(200, 40, 100)
x3_unwanted = np.full_like(x1_unwanted, 150)
x2_unwanted = constraint1(x1_unwanted)

#plot the unwanted region surface
ax.plot_surface(x1_unwanted.reshape(-1,1), x2_unwanted.reshape(-1,1), x3_unwanted.reshape(-

#plot the optimal solution point
optimal_x1 = 200.0
optimal_x2 = 300.0
optimal_x3 = 0.0
ax.scatter(optimal_x1, optimal_x2, optimal_x3, color='red', s=100, label='Optimal Solution')

ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title('Feasible Region with Optimal Solution')
plt.show()
```

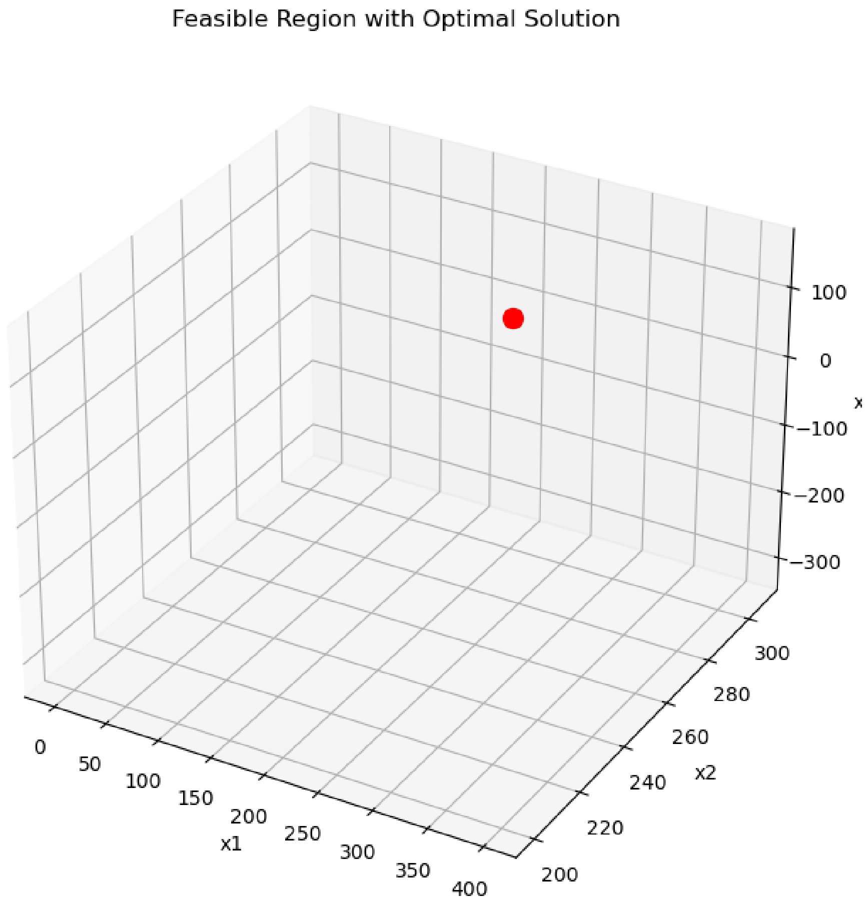

Figure 5: Graph for number five

```
    NO. 6 FINANCIAL PORTFOLIO OPTIMIZATION
    from pulp import LpVariable,LpMaximize,LpProblem
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model=LpProblem("Financial_Portfolio_Optimization",sense= LpMaximize)
#LpVariables
x1=LpVariable("x1",0)
x2=LpVariable("x2",0)
x3=LpVariable("x3",0)
#Objective function
model+=0.08*x1 + 0.1*x2 + 0.12*x3
#Constraints
model+=2*x1 + 3*x2 + x3 <=10000,"maximum_budget_constraint"
model+=x1>=2000,"minimize_x1"
model+=x2>=1500,"minimize_x2"
model+=x3>=1000,"minimize_x3"
#solving
model.solve()
#results
optimal_x1=x1.varValue
optimal_x2=x2.varValue
optimal_x3=x3.varValue
optimal_value=model.objective.value()
print("optimal solution")
print("x1",optimal_x1)
print("x2",optimal_x2)
print("x3",optimal_x3)
print("z:",optimal_value)
optimal solution
x1 2000.0
x2 1500.0
x3 1500.0
z: 490.0

\[CODES FOR THE GRAPH\]
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D

#Define the constraints
def constraint1(x1):
    return (10000 - 2 * x1)/3

def constraint2(x1):
    return (10000 - 2 * x1)/3
```

```
def constraint3(x1):
    return (10000 - 2 * x1)/3

#create arrays of x1 values for plotting
x1_values = np.linspace(0, 5000, 1000)

#calculate corresponding x2 and x3 values based on the constraints
x2_values = constraint1(x1_values)
x3_values = constraint2(x1_values)

#plot the constraints
fig = plt.figure(figsize=(10, 8))
ax = fig.add_subplot(111,projection='3d')
ax.plot(x1_values, x2_values, x3_values, color='b', alpha=0.3)

#plot the feasible region surface
x1,x3 = np.meshgrid(x1_values, x3_values)
x2 = constraint1(x1_values)
ax.plot_surface(x1, x2, x3, color='gray', alpha=0.5, label='Feasible Region')



#plot the optimal solution point
optimal_x1 = 2000 # optimal value for x1
optimal_x2 = 1500 # optimal value for x2
optimal_x3 = 1500 # optimal value for x3
ax.scatter(optimal_x1, optimal_x2, optimal_x3, color='red', s=100, label='Optimal Solution'

ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title('Feasible Region with Optimal Solution')
plt.show()
```

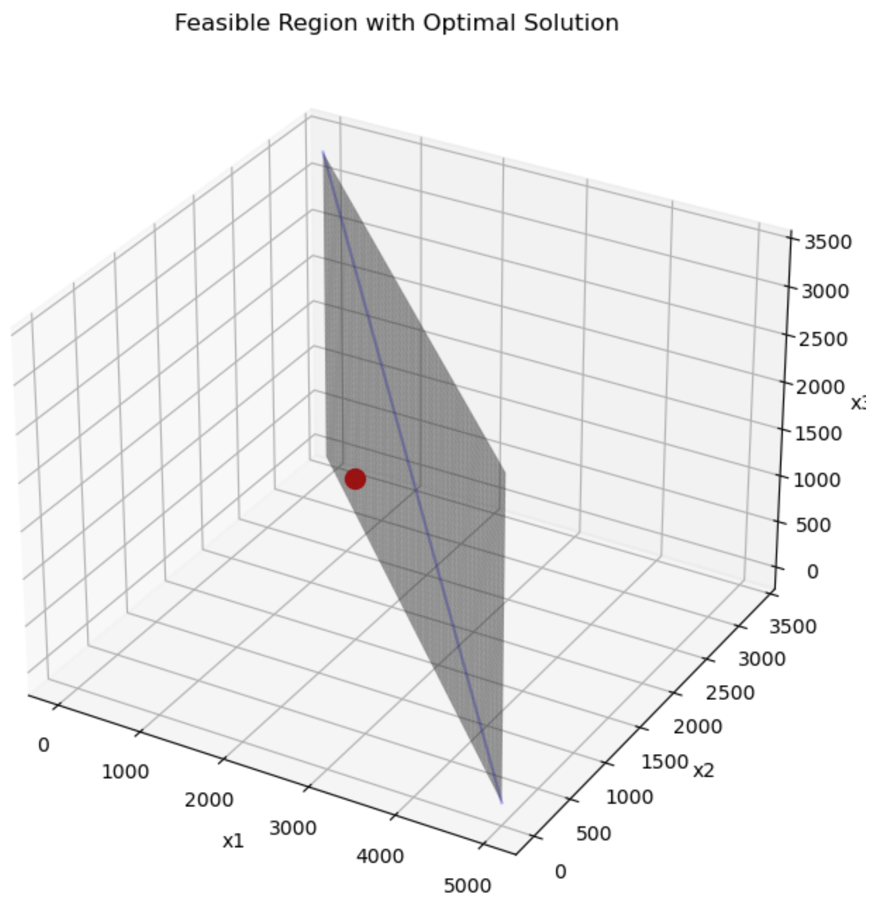

Figure 6: Graph for number six

```
No. 7 DIET OPTIMIZATION
from pulp import LpProblem,LpMinimize,LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="Diet_Optimization",sense=LpMinimize)
#variables
x1=LpVariable("x1",0)
x2=LpVariable("x2",0)
#objective function
model+= 3*x1 + 2*x2
#Constraints
model+= 2*x1 + x2 >= 20,"Proteins Constraint"
model+= 3*x1 + 2*x2 >= 25,"Vitamins Constraint"
#Solving
model.solve()
#resluts
optimal_x1=x1.varValue
optimal_x2=x2.varValue
optimal_value = model.objective.value()
print("optimal solution:")
print("x1:",optimal_x1)
print("x2:",optimal_x2)
print("z:",optimal_value)
optimal solution:
x1: 10.0
x2: 0.0
z: 30.0

\[CODES FOR GRAPH\]
import matplotlib.pyplot as plt
import numpy as np
#define x array
x=np.linspace(0,16,15000)
#convert constraints to inequalities
x1=(20-2*x)
x2=(25-3*x)/2
#plot constraints
plt.plot(x,x1,label="2*x1 + x2 >=20")
plt.plot(x,x2,label="3*x1 + 2*x2 >=25")
#define the feasible region
x3 =np.maximum.reduce([x1,x2])
plt.fill_between(x,x3,25,color="pink",label="feasible region",alpha=0.3)
# define limits
plt.xlim(0,11)
plt.ylim(0,25)
plt.grid(True)
plt.xlabel ("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```

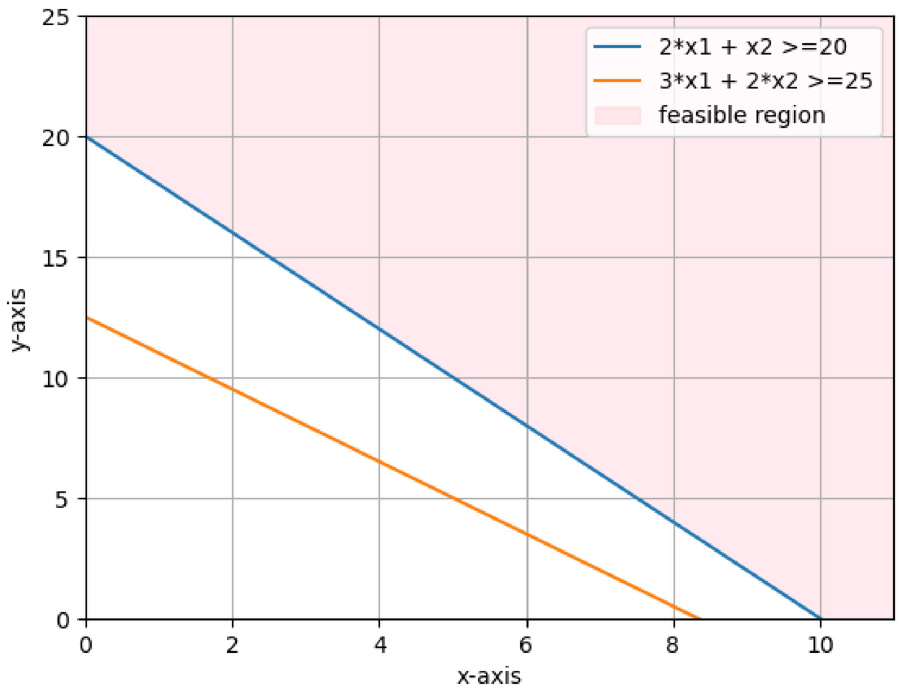

Figure 7: Graph for number seven(7)

```
    NO. 8 PRODUCTION PLANNING
    from pulp import LpProblem,LpMaximize,LpVariable
import matplotlib.pyplot as plt
import numpy as np
#LpProblem
model = LpProblem(name="Production_Planning",sense=LpMaximize)
#variables
x1=LpVariable("x",0)
x2=LpVariable("y",0)
#objective function
model+= 5*x1 + 3*x2
#Constraints
model+= 2*x1 + 3*x2 <= 60,"Labour Constraint"
model+= 4*x1 + 2*x2 <= 80,"Tenant 2 Constraint"
#Solving
model.solve()
#resluts
optimal_x1=x1.varValue
optimal_x2=x2.varValue
optimal_value = model.objective.value()
print("optimal solution:")
print("x1:",optimal_x1)
print("x2:",optimal_x2)
print("z:",optimal_value)
optimal solution:
x1: 15.0
x2: 10.0
z: 105.0

\[CODES FOR GRAPH\]
import matplotlib.pyplot as plt
import numpy as np
#define x array
x=np.linspace(0,100,15000)
#convert constraints to inequalities
x1=(60-2*x)/3
x2=(80-4*x)/2
#plot constraints
plt.plot(x,x1,label="2*x1 + 3x2 <=60")
plt.plot(x,x2,label="4*x1 + 2*x2 <=80")
#define the feasible region
x3 =np.minimum.reduce([x1,x2])
plt.fill_between(x,x3,0, color="green", alpha=0.3, label="feasible region")
# define limits
plt.xlim(0,40)
plt.ylim(0,45)
plt.grid(True)
plt.xlabel ("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```

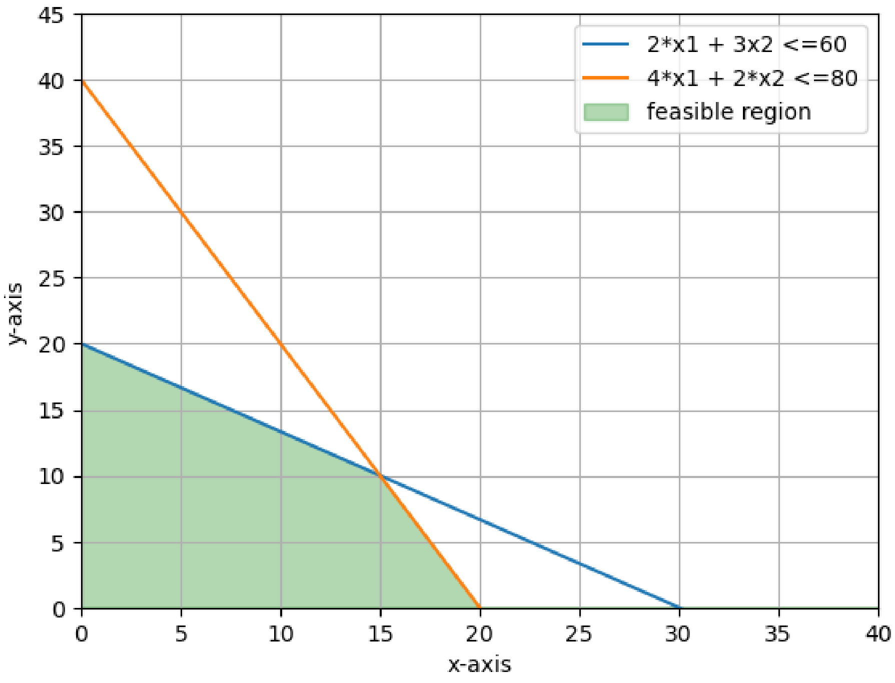

Figure 8: Graph for number eight