MOUNTAINS OF THE MOON UNIVERSITY

FACULTY OF SCIENCE TECHNOLOGY AND INNOVATION

DEPARTMENT OF COMPUTER SCIENCE

NAME:	**MPAGI DERRICK BRAIR**
REG NO.	**2023/U/MMU/BCS/00101**
PAPER CODE	**BCS 1201**
COURSE	**LINAR PROGRAMMING**
LECTURER NAME	**MR OCEN SAMMUEL**
YEAR	**ONE**
SEMISTER	**TWO**

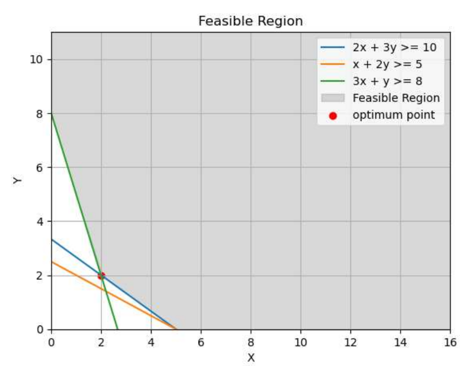
COURSE WORK

NO1

```
# importing neccesary libaries
from pulp import LpVariable, LpProblem, LpMinimize
import numpy as np
import matplotlib.pyplot as plt

# creating linear problem
problem = LpProblem(name="cost-minimizing", sense=LpMinimize)

# Decision Variables
x = LpVariable(name="x", lowBound=0)
y = LpVariable(name="y", lowBound=0)

# define objective function
problem += 4*x + 5*y, "objective"

# define constraints
problem += 2*x + 3*y>=10, "CPU"
problem += x + 2*y >= 5, "Memory"
problem += 3*x + y >= 8, "Storage"

# solve
problem.solve()

# display results
print("OPTIMUM SOLUTION")
print(f"X: {x.varValue}")
print(f"Y: {y.varValue}")
print(f"Minimum cost: {problem.objective.value()}")

#...........THE GRAPH...........
#  storing optimum points
x_min = x.varValue
y_min = y.varValue
# x array
x = np.linspace(0, 30, 30)

# Constraints (converted to inequalities)
y1 = (10 - 2*x) / 3.0
y2 = (5 - x) / 2.0
y3 = 8 - 3*x

# Plot constraints
plt.plot(x, y1, label="2x + 3y >= 10")
```

```
plt.plot(x, y2, label="x + 2y >= 5")
plt.plot(x, y3, label="3x + y >= 8")

# Plotting the feasible region
y4 = np.maximum.reduce([y1, y2, y3]) # Upper boundary of the feasible region
plt.fill_between(x, y4, 11, color='gray', alpha=0.3, label="Feasible Region")

# plotting the optimum point
plt.scatter(x_min,y_min, color="red", label="optimum point")

# Axis limits and labels
plt.xlim(0, 16)
plt.ylim(0, 11)
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.title("Feasible Region")
plt.grid()

plt.show()
```

```
OPTIMUM SOLUTION
X: 2.0
Y: 2.0
Minimum cost: 18.0
```

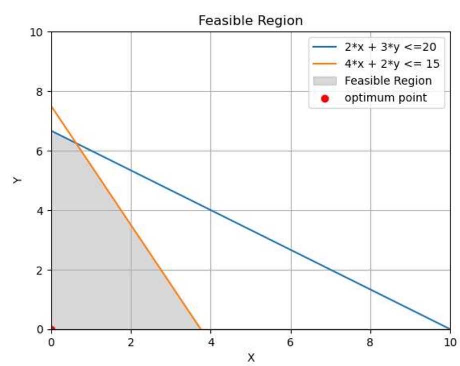


NO2

```
# importing neccesary libaries
from pulp import LpVariable, LpProblem, LpMinimize
import numpy as np
import matplotlib.pyplot as plt

# creating linear problem
problem = LpProblem(name="cost-minimizing", sense=LpMinimize)

# Decision Variables
x = LpVariable(name="x", lowBound=0)
y = LpVariable(name="y", lowBound=0)


# define objective function
problem += 5*x + 4*y, "objective"

# define constraints
problem += 2*x + 3*y <=20, "Server 1 Capacity"
problem += 4*x + 2*y <= 15, "Server 2 Capacity"

# solve
problem.solve()

# display results
print("OPTIMUM SOLUTION")
print(f"X: {x.varValue}")
print(f"Y: {y.varValue}")
print(f"Minimum cost: {problem.objective.value()}")

 #........The graph......

 #  storing optimum points
x_min = x.varValue
y_min = y.varValue

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

# Constraints (converted 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")

# Plotting the feasible region
y3 = np.minimum.reduce([y1, y2]) # Upper boundary of the feasible region

plt.fill_between(x, y3, 0, color='gray', alpha=0.3, label="Feasible Region")

# plotting the optimum point
plt.scatter(x_min,y_min, color="red", label="optimum point")

# Axis limits and labels
plt.xlim(0, 10)
plt.ylim(0, 10)
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.title("Feasible Region")
plt.grid()
plt.show()
```

```
OPTIMUM SOLUTION
X: 0.0
Y: 0.0
Minimum cost: 0.0
```

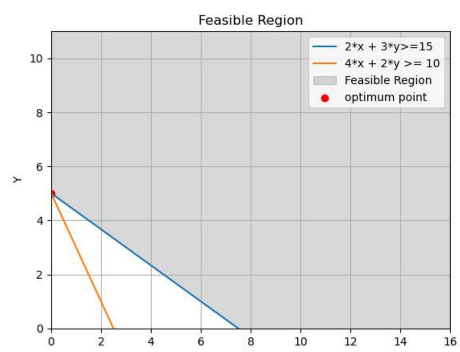


NO3

```
# importing neccesary libaries
from pulp import LpVariable, LpProblem, LpMinimize
import numpy as np
import matplotlib.pyplot as plt

# creating linear problem
problem = LpProblem(name="cost-minimizing", sense=LpMinimize)

# Decision Variables
x = LpVariable(name="x", lowBound=0)
y = LpVariable(name="y", lowBound=0)

# define objective function
problem += 3*x + 2*y, "objective"

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

# solve
problem.solve()

# display results
print("OPTIMUM SOLUTION")
print(f"X: {x.varValue}")
print(f"Y: {y.varValue}")
print(f"Minimum cost: {problem.objective.value()}")

 #...........THE GRAPH...........

#       storing minimum point
x_min = x.varValue
y_min = y.varValue

x = np.linspace(0, 16, 2000)

# Constraints (converted 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")

# Plotting the feasible region
y3 = np.maximum.reduce([y1, y2]) # Upper boundary of the feasible region
plt.fill_between(x, y3, 11, color='gray', alpha=0.3, label="Feasible Region")

# plotting the optimum point
plt.scatter(x_min,y_min, color="red", label="optimum point")

# Axis limits and labels
plt.xlim(0, 16)
plt.ylim(0, 11)
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.title("Feasible Region")
plt.grid()

plt.show()
```

```
OPTIMUM SOLUTION
X: 0.0
Y: 5.0
Minimum cost: 10.0
```

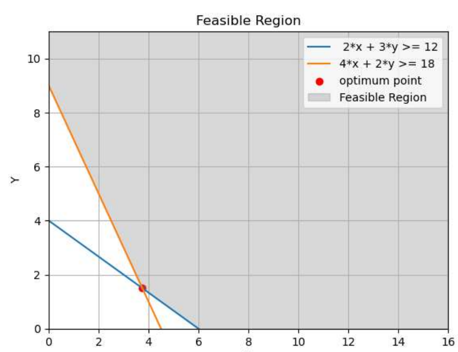


NO4

```
# importing neccesary libaries
from pulp import LpVariable, LpProblem, LpMinimize
import numpy as np
import matplotlib.pyplot as plt

# creating linear problem
problem = LpProblem(name="cost-minimizing", sense=LpMinimize)

# Decision Variables
x = LpVariable(name="x", lowBound=0)
y = LpVariable(name="y", lowBound=0)

# define objective function
problem += 5*x + 4*y, "objective"

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

# solve
problem.solve()

# display results
print("OPTIMUM SOLUTION")
print(f"X: {x.varValue}")
print(f"Y: {y.varValue}")
print(f"Minimum cost: {problem.objective.value()}")

#...........THE GRAPH...........

#      storing minimum point
x_min = x.varValue
y_min = y.varValue

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

# Constraints (converted 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")

# Plotting the feasible region
y3 = np.maximum.reduce([y1, y2]) # Upper boundary of the feasible region

# plotting the optimum point
plt.scatter(x_min,y_min, color="red", label="optimum point")

# plt.fill_between(x, y3, 0, color='gray', alpha=0.3, label="Feasible Region")
plt.fill_between(x, y3, 11, color='gray', alpha=0.3, label="Feasible Region")

# Axis limits and labels
plt.xlim(0, 16)
plt.ylim(0, 11)
plt.xlabel("X")
plt.ylabel("Y")
plt.legend()
plt.title("Feasible Region")
plt.grid()

plt.show()
```

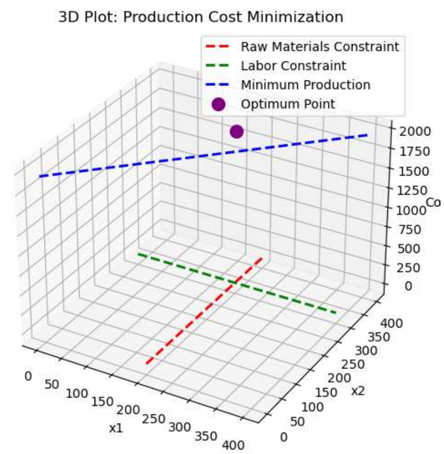
```
OPTIMUM SOLUTION
X: 3.75
Y: 1.5
Minimum cost: 24.75
```



NO5

```
# importing neccesary libaries
from pulp import LpVariable, LpProblem, LpMinimize
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# creating linear problem
problem = LpProblem(name="production-minimizing", sense=LpMinimize)

# Decision Variables
x1 = LpVariable(name="x", lowBound=0)
x2 = LpVariable(name="y", lowBound=0)
x3 = LpVariable(name="z", lowBound=0)


# define objective function
problem += 5*x1 + 3*x2 + 4*x3, "objective"

# define constraints
problem += 2*x1 + 3*x2 + x3 <= 1000, "Raw_materials"
problem += 4*x1 + 2*x2 + 5*x3 <= 120, "Labour"
problem += x1 >=200
problem += x2 >=300
problem += x3 >=150

# solve
problem.solve()

# display results
print("OPTIMUM SOLUTION")
print(f"X: {x1.varValue}")
print(f"Y: {x2.varValue}")
print(f"Z: {x3.varValue}")
print(f"Minimum cost: {problem.objective.value()}")

# plotting the graph

# Create a meshgrid for x1, x2, and x3
x1_vals = np.linspace(0, 400, 50)
x2_vals = np.linspace(0, 400, 50)
x1_grid, x2_grid = np.meshgrid(x1_vals, x2_vals)

# Calculate the corresponding z-values (objective function)
```

```
z_vals = 5 * x1_grid + 3 * x2_grid + 4 * (1950 - x1_grid - x2_grid)

# Create the 3D plot
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Plot the feasible region (constraints)
ax.plot([200, 200], [0, 400], [0, 0], color='red', linestyle='--', linewidth=2,
ax.plot([0, 400], [300, 300], [0, 0], color='green', linestyle='--', linewidth=2
ax.plot([0, 400], [0, 400], [1950, 1950], color='blue', linestyle='--', linewidt

# Highlight the optimum point
optimum_x1 = 200
optimum_x2 = 300
optimum_z = 1950
ax.scatter(optimum_x1, optimum_x2, optimum_z, color='purple', s=100, label='Optim

# Set labels and title
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title('Production Cost Minimization')

# Add a legend
ax.legend()

# Show the plot
plt.show()


OPTIMUM SOLUTION
X: 200.0
Y: 300.0
Z: 0.0
Minimum cost: 1900.0
```

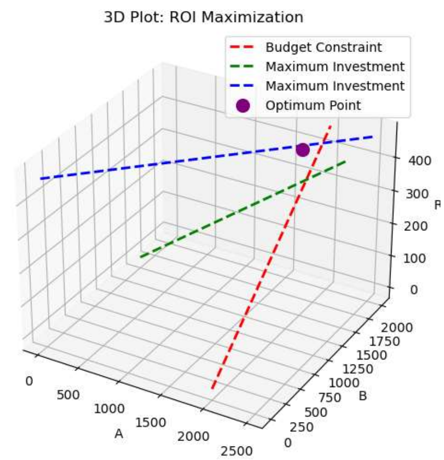


NO6

```
# importing neccesary libaries
from pulp import LpVariable, LpProblem, LpMaximize
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# creating linear problem
problem = LpProblem(name="production-maximizing", sense=LpMinimize)

# Decision Variables
x1 = LpVariable(name="A", lowBound=0)
x2 = LpVariable(name="B", lowBound=0)
x3 = LpVariable(name="C", lowBound=0)


# define objective function
problem += 0.08*x1 + 0.1*x2 + 0.12*x3, "objective"


# define constraints
problem += 2*x1 + 3*x2 + x3 <= 10000, "budget_for_investment"
problem += x1 >=2000
problem += x2 >=1500
problem += x3 >=1000

# solve
```

```python
problem.solve()

# display results
print("OPTIMUM SOLUTION")
print(f"A: {x1.varValue}")
print(f"B: {x2.varValue}")
print(f"C: {x3.varValue}")
print(f"Maximum Return on Investment: {problem.objective.value()}")

# Create a meshgrid for A, B, and C
A_vals = np.linspace(0, 2500, 50)
B_vals = np.linspace(0, 2000, 50)
A_grid, B_grid = np.meshgrid(A_vals, B_vals)

# Calculate the corresponding z-values (ROI function)
C_vals = (10000 - 2 * A_grid - 3 * B_grid)  # Budget constraint
ROI_vals = 0.08 * A_grid + 0.1 * B_grid + 0.12 * C_vals

# Create the 3D plot
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Plot the feasible region (constraints)
ax.plot([2000, 2000], [0, 2000], [0, 470], color='red', linestyle='--', linewidt
ax.plot([0, 2500], [1500, 1500], [0, 470], color='green', linestyle='--', linewi
ax.plot([0, 2500], [0, 2000], [470, 470], color='blue', linestyle='--', linewidt

# Highlight the optimum point
optimum_A = 2000
optimum_B = 1500
optimum_ROI = 470
ax.scatter(optimum_A, optimum_B, optimum_ROI, color='purple', s=100, label='Opti

# Set labels and title
ax.set_xlabel('A')
ax.set_ylabel('B')
ax.set_zlabel('ROI')
ax.set_title('3D Plot: ROI Maximization')

# Add a legend
ax.legend()

# Show the plot
plt.show()
```

```
OPTIMUM SOLUTION
A: 2000.0
B: 1500.0
C: 1000.0
Maximum Return on Investment: 430.0
```

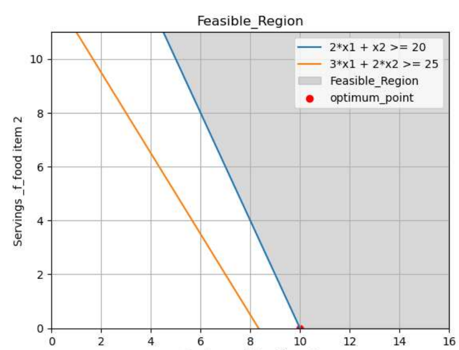


NO7

```
# importing necessary libraries
from pulp import LpVariable, LpProblem, LpMinimize
import numpy as np
import matplotlib.pyplot as plt

# creating linear problem
problem = LpProblem(name="Diet Minimize", sense=LpMinimize)

# Decision Variables
x1 = LpVariable(name="Servings of food item 1", lowBound=0)
x2 = LpVariable(name="Servings of food item 2", lowBound=0)

# define objective function
problem += 3*x1 + 2*x2, "objective"

# define constraints
problem += 2*x1 + x2 >= 20, "Proteins"
problem += 3*x1 + 2*x2 >= 25, "Vitamins"

# solve
problem.solve()

# display results
print("DIET OPTIMUM SOLUTION")
print(f"X: {x1.varValue}")
print(f"Y: {x2.varValue}")
print(f"Minimum cost: {problem.objective.value()}")

 #...........THE GRAPH...........

#       storing minimum point
x_min = x1.varValue
y_min = x2.varValue

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

# Constraints
x2_proteins = 20 - 2*x
x2_vitamins = (25 - 3*x) / 2
```

```
# Plot constraints
plt.plot(x, x2_proteins, label="2*x1 + x2 >= 20")
plt.plot(x, x2_vitamins, label="3*x1 + 2*x2 >= 25")

# Plotting the feasible region
y3 = np.maximum.reduce([x2_proteins, x2_vitamins, np.zeros_like(x)])
# Upper boundary of the feasible region
plt.fill_between(x, y3, 11, color='gray', alpha=0.3, label="Feasible_Region")

# plotting the optimum point
plt.scatter(x_min,y_min, color="red", label="optimum_point")

# Axis limits and labels
plt.xlim(0, 16)
plt.ylim(0, 11)
plt.xlabel("Servings_of_food item 1")
plt.ylabel("Servings _f_food item 2")
plt.legend()
plt.title("Feasible_Region")
plt.grid()

plt.show()
```

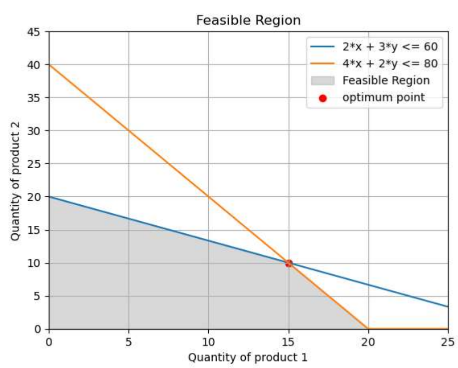
```
DIET OPTIMUM SOLUTION
X: 10.0
Y: 0.0
Minimum cost: 30.0
```



NO8

```
# importing necessary libraries
from pulp import LpVariable, LpProblem, LpMaximize
import numpy as np
import matplotlib.pyplot as plt


# creating linear problem
problem = LpProblem(name="production profit maximization", sense=LpMaximize)

# Decision Variables
x1 = LpVariable(name="Quantity of product 1", lowBound=0)
x2= LpVariable(name="Quantity of product 2", lowBound=0)


# define objective function
problem += 5*x1 + 3*x2, "objective"


# define constraints
problem += 2*x1 + 3*x2  <= 60, "labour"
problem += 4*x1 + 2*x2  <= 80, "raw_materials"


# solve
problem.solve()

# display results
print("Production Profit Maximization")
print(f"Quantity of product 1: {x1.varValue}")
print(f"Quantity of product 2: {x2.varValue}")

print(f"Maximum Profit : {problem.objective.value()}")

 #...........THE GRAPH...........

#      storing minimum point
x_min = x1.varValue
y_min = x2.varValue

x = np.linspace(0, 50, 2000)

# Constraints
```

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

# Ensure constraints are non-negative
y1 = np.maximum(y1, 0)
y2 = np.maximum(y2, 0)

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

# Plotting the feasible region
y3 = np.minimum(y1, y2)  # Feasible region is where both constraints are satisfi
plt.fill_between(x, y3, 0, color='gray', alpha=0.3, label="Feasible Region")

# plotting the optimum point
plt.scatter(x_min,y_min, color="red", label="optimum point")

# Axis limits and labels
plt.xlim(0, 25)
plt.ylim(0, 45)
plt.xlabel("Quantity of product 1")
plt.ylabel("Quantity of product 2")
plt.legend()
plt.title("Feasible Region")
plt.grid()

plt.show()

Production Profit Maximization
Quantity of product 1: 15.0
Quantity of product 2: 10.0
Maximum Profit        : 105.0
```


Feasible Region
2*x + 3*y <= 60
4*x + 2*y <= 80
Feasible Region
optimum point
Quantity of product 2
Quantity of product 1
0
5
10
15
20
25
30
35
40
45