1. **LPP** - Max z = 7x1 + 10x2
   Subject to
   c1 = 5x1 + 6x2 <= 3600
   c2 = x1 +2x2 <= 960
   c3 = x1 <= 500
   c4 = x2 <= 500
   x1, x2 >= 0

Max z = 7x1 + 10x2			Objective	x1	x2			
subject to				7	10			
c1 = 5x1 + 6x2 <= 3600			z	360	300	5520		
c2 = x1 + 2x2 <=960			c1	5	6	3600	<=	3600
c3 = x1<=500			c2	1	2	960	<=	960
c4 = x2 <= 500			c3	1	0	360	<=	500
x1, x2 >=0			c4	0	1	300	<=	500

Sensitivity Analysis Report-

Microsoft Excel 16.0 Sensitivity Report
Worksheet: [Book1]Sheet1
Report Created: 10-10-2023 13:52:21

Variable Cells

Cell	Name	Final Value	Reduced Cost	Objective Coefficient	Allowable Increase	Allowable Decrease
E3	z x1	360	0	7	1.333333333	2
F3	z x2	300	0	10	4	1.6

Constraints

Cell	Name	Final Value	Shadow Price	Constraint R.H. Side	Allowable Increase	Allowable Decrease
G4	c1	3600	1	3600	280	720
G5	c2	960	2	960	160	93.33333333
G6	c3	360	0	500	1E+30	140
G7	c4	300	0	500	1E+30	200

b.) Cost efficient sensitivity analysis –

The cost of Baseball can be increased by $1.333333333 and has an allowable decrease of $2.

And the cost of Softball can be increased by $4 and it has an allowable decrease of $1.6.

c.) Right hand side sensitivity analysis –

Out of 500 Baseballs, we are manufacturing only 360 balls. Hence, the allowable decrease is 140.

Out of 500 Softballs, we are manufacturing only 300 balls. Therefore, the allowable decrease is 200.

We are utilizing all the cowhide (3600 sq. feet), and there's an allowable increase of 280, and can be decreased by 720. It has a shadow price of $1.

We are utilizing all the time (960 minutes), and there's an allowable increase of 160, and can be decreased by 93.33333333. It has a shadow price of $2.

2.

```
#importing libraries
import numpy as np
import matplotlib.pyplot as plt
import scipy as sp
```

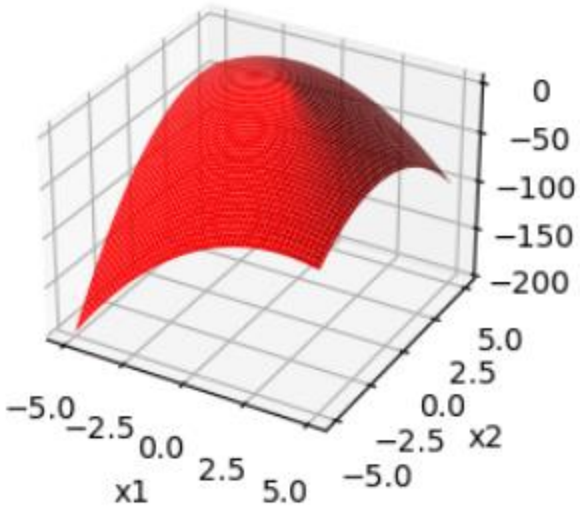
$f(x1, x2) = 4x1 + 6x2 - 2x1^2 - 2x1x2 - 2x2^2$

```
def f(x1, x2):  #defining the function
    return 4*x1 + 6*x2 - 2*x1**2 - 2*x1*x2 - 2*x2**2

x1 = np.linspace(-5, 5, 100) #creating meshgrid
x2 = np.linspace(-5, 5, 100)
X1, X2 = np.meshgrid(x1, x2)
Z = f(X1, X2)

fig = plt.figure()  #plotting the graph
ax = plt.subplot(1, 2, 1, projection='3d')
ax.plot_surface(X1, X2, Z, color = 'red')

ax.set_xlabel('x1')  #labelling x and y axes
ax.set_ylabel('x2')
#ax.set_zlabel('f(x1, x2)')
plt.show()
```




```python
def gradient(x1, x2):
    dx1 = 4 - 4*x1 - 2*x2  #differentiating wrt x1
    dx2 = 6 - 2*x1 - 4*x2  #differentiating wrt x2
    return np.array([dx1, dx2])

x = np.array([0, 0])
lr = 0.1  #learning rate = 0.1
max_itns = 100

for i in range(max_itns):
    grad = gradient(x[0], x[1])
    x = x + lr * grad

max_val = f(x[0], x[1])
max_x1, max_x2 = x[0], x[1]

print("Max. value:", max_val)
print("Optimal (x1, x2):", (max_x1, max_x2))
```

```
Max. value: 4.666666666666667
Optimal (x1, x2): (0.3333333334351851, 1.3333333332314816)
```