```
In [1]: from pulp import *
```

Question 1

1.1 `c <= 0; a <= 0; b <= 0; all others free`

A tableau is optimal if and only if all coefficients of the variables are nonpositive.

1.2 `c > 0; d >= max(0, 5e/6); all others free`

The selection of optimal exiting variables is based on the min ratio rule. By this rule, this means to exit on `x3` the constraint `5 / d >= 6 /e` must be sastified. We equalize the constraint by transforming it to `30/6d >= 30/5e` therefore it can be reduced to `6d >= 5e` and simplified to `d >= 5e/6`. We add a `0` constraint in the event `e` is negative.

1.3 `c > 0; e <= 0; d <= 0; all others free`

A problem is unbounded if all coefficients of the entering variable column are nonpositive. We select entering variables which are positive.

Question 2

2.1

```
max z = x[1] + 2*x[2] + 3*x[3]
s.t.
  x[1] +   x[2] +   x[3] + s[1] = 16
3*x[1] + 2*x[2] + 2*x[3]        = 26
  x[1] +          x[3] - s[2] = 10
x[1], x[2], x[3], s[1], s[2]   >= 0
```

2.2

```
min. z = y[1] + y[2]
s.t.
  x[1] +   x[2] +   x[3] + s[1]        = 16
3*x[1] + 2*x[2] + 2*x[3]        + y[1] = 26
  x[1] +          x[3] - s[2] + y[2] = 10
x[1], x[2], x[3], s[1], s[2], y[1], y[2] >= 0
```

2.3

Form initial tableau in canonical form

```
   -z  x1  x2  x3  s1  s2  y1  y2   RHS
r1  1   5   3   4   1  -1   0   0    52
r2  0   1   1   1   1   0   0   0    16
r3  0   3   2   2   0   0   1   0    26
```

```
r4  0   1   0   1   0  -1   0   1    10
```

enter *x3*; exit *y2*;

```
   -z  x1  x2  x3  s1  s2  y1  y2   RHS
r1  1   1   3   0   1   3   0  -4    12 ; = r1  - 4*r4
r2  0   0   1   0   1   1   0  -1     6 ; = r2  - r4
r3  0   1   2   0   0   2   1  -2     6 ; = r3  - 2*r4
r4  0   1   0   1   0  -1   0   1    10
```

enter *x1*; exit *y1*;

```
   -z  x1  x2  x3  s1  s2  y1  y2   RHS
r1  1   0   1   0   1   1  -1  -2     6 ; r1 = r1 - r3
r2  0   0   1   0   1   1   0  -1     6
r3  0   1   2   0   0   2   1  -2     6
r4  0   0  -2   1   0  -3  -1   3     4 ; r4 = r4 - r3
```

enter *x2*; exit *x1*;

```
   -z   x1   x2   x3   s1   s2   y1   y2    RHS
r1  1 -0.5    0    0    1    0 -1.5   -1      3 ; r1 = r1 - 0.5*r3
r2  0 -0.5    0    0    1    0 -0.5    0      3 ; r2 = r2 - 0.5*r3
r3  0  0.5    1    0    0    1  0.5   -1      3 ; r3 = 0.5*r3
r4  0    1    0    1    0   -1    0    1     10 ; r4 = r4 + r3
```

pivot *s1*;

```
   -z   x1   x2   x3   s1   s2   y1   y2    RHS
r1  1    0    0    0    0    0   -1   -1      0 ; r1 = r1 - r2
r2  0 -0.5    0    0    1    0 -0.5    0      3 ;
r3  0  0.5    1    0    0    1  0.5   -1      3 ;
r4  0    1    0    1    0   -1    0    1     10 ;
```

Optimal solution reached, RHS = 0 for z-row, feasible basis achieved, proceed to phase 2.

In [2]:
```
# verification of phase 1

prob = LpProblem("Phase 1",LpMinimize)
x1=LpVariable("x1", 0, None)
x2=LpVariable("x2", 0, None)
x3=LpVariable("x3", 0, None)
s1=LpVariable("s1", 0, None)
s2=LpVariable("s2", 0, None)
y1=LpVariable("y1", 0, None)
y2=LpVariable("y2", 0, None)

prob += y1 + y2
prob += x1   + x2   +   x3 + s1      == 16
```

```
prob += 3*x1 + 2*x2 + 2*x3      + y1 == 26
prob += x1   +           x3 - s2 + y2 == 10


prob.solve()
print("Optimal Value for Phase 1 Problem")
print("z = {}".format(value(prob.objective)))
```

2.4

```
   -z   x1   x2   x3   s1   s2    RHS
r1  1    1    2    3    0    0      0 ;
r2  0 -0.5    0    0    1    0      3 ;
r3  0  0.5    1    0    0    1      3 ;
r4  0    1    0    1    0   -1     10 ;
```

pivot *x2*;

```
   -z   x1   x2   x3   s1   s2    RHS
r1  1    0    0    3    0   -2     -6 ; r1 = r1 - 2*r3
r2  0 -0.5    0    0    1    0      3 ;
r3  0  0.5    1    0    0    1      3 ;
r4  0    1    0    1    0   -1     10 ;
```

pivot *x3*;

```
   -z   x1   x2   x3   s1   s2    RHS
r1  1   -3    0    0    0    1    -36 ; r1 = r1 - 3*r4
r2  0 -0.5    0    0    1    0      3 ;
r3  0  0.5    1    0    0    1      3 ;
r4  0    1    0    1    0   -1     10 ;
```

enter *s2*; exit *x2*;

```
   -z   x1   x2   x3   s1   s2    RHS
r1  1 -3.5   -1    0    0    0    -39 ; r1 = r1 - r3
r2  0 -0.5    0    0    1    0      3 ;
r3  0  0.5    1    0    0    1      3 ;
r4  0  1.5    1    1    0    0     13 ; r4 = r4 + 3
```

Tableau is now optimal, the objective value is 39 and the optimal vector is `(0, 0, 13, 3, 3)`

```
In [9]: # verification for phase 2

        prob = LpProblem("Phase 2",LpMaximize)

        prob +=       x1    + 2*x2 + 3*x3
        prob += -0.5*x1    + 0*x2 + 0*x3 + 1*s1 + 0*s2 == 3
        prob +=  0.5*x1    + 1*x2 + 0*x3 + 0*s1 + 1*s2 == 3
```

```
        prob +=      x1    + 0*x2 + 1*x3 + 0*s1 - 1*s2 == 10

        prob.solve()
        phase2val = value(prob.objective)

        prob = LpProblem("Original Problem",LpMaximize)

        prob +=   x1 + 2*x2 + 3*x3
        prob += 1*x1 + 1*x2 + 1*x3 <= 16
        prob += 3*x1 + 2*x2 + 2*x3 == 26
        prob += 1*x1 + 0*x2 + 1*x3 >= 10
        prob.solve()
        val = value(prob.objective)

        print("Optimal value for phase 2 problem")
        print("z = {}".format(phase2val))
        print("Optimal value for original problem")
        print("z = {}".format(val))
        print("Is phase 2 equal original? {}".format(phase2val == val))
```

```
Optimal value for phase 2 problem
z = 39.0
Optimal value for original problem
z = 39.0
Is phase 2 equal original? True
```

Question 3

3.1

```
v w = 0  1  2  3  4  5  6  7  8  9 10 11
0     0  0  0  0  0  0  0  0  0  0  0  0
1     0  0  0  0  0  4  4  4  4  4  4  4
2     0  0  0  0  0  4  9  9  9  9  9 13
3     0  0  0  0  7  7  9  9  9 11 16 16
4     0  0  0  6  7  7  9 13 13 15 16 16
5     0  0  0  6  7  7  9 13 13 15 19 20
```

The optimal solution is 20 with the optimal vector `(0, 0, 1, 0, 1)`

```
In [4]: # verification for 3.1

        import pprint
        n = 5
        W = 11

        vs = [4, 9, 7, 6, 13]
```

```python
ws = [5, 6, 4, 3, 7]

V = [[0 for i in range(W + 1)] for j in range(n + 1)]

for i in range(1, n + 1):
    v_i = vs[i - 1]
    w_i = ws[i - 1]
    for w in range(1, W + 1):
        if w < w_i:
            V[i][w] = V[i - 1][w]
        else:
            V[i][w] = max(V[i - 1][w], v_i + V[i - 1][w - w_i])
pprint.pprint(V)
```

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 4, 4, 4, 4, 4, 4, 4],
 [0, 0, 0, 0, 0, 4, 9, 9, 9, 9, 9, 13],
 [0, 0, 0, 0, 7, 7, 9, 9, 9, 11, 16, 16],
 [0, 0, 0, 6, 7, 7, 9, 13, 13, 15, 16, 16],
 [0, 0, 0, 6, 7, 7, 9, 13, 13, 15, 19, 20]]
```

In [5]:
```python
# solution for 3.1

prob = LpProblem("Knapsack without Replacement",LpMaximize)
x1=LpVariable("x1", 0, 1, LpInteger)
x2=LpVariable("x2", 0, 1, LpInteger)
x3=LpVariable("x3", 0, 1, LpInteger)
x4=LpVariable("x4", 0, 1, LpInteger)
x5=LpVariable("x5", 0, 1, LpInteger)

prob += 4*x1 + 9*x2 + 7*x3 + 6*x4 + 13*x5
prob += 5*x1 + 6*x2 + 4*x3 + 3*x4 + 7*x5 <= 11

prob.solve()
val = value(prob.objective)
print("Optimal Value for Knapsack without Replacement")
for v in prob.variables():
    print("{} = {}".format(v.name, v.varValue))
print("z = {}".format(val))
```

```
Optimal Value for Knapsack without Replacement
x1 = 0.0
x2 = 0.0
x3 = 1.0
x4 = 0.0
x5 = 1.0
z = 20.0
```

3.2

v w =	0	1	2	3	4	5	6	7	8	9	10	11
0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	4	4	4	4	4	8	8
2	0	0	0	0	0	4	9	9	9	9	9	13
3	0	0	0	0	7	7	9	9	14	14	16	16
4	0	0	0	6	7	7	12	12	14	18	19	20
5	0	0	0	6	7	7	12	13	14	18	19	20

The optimal solution is 20 with the optimal vector `(0, 0, 1, 0, 1)`

In [6]:
```
# verification for 3.2

n = 5
W = 11

vs = [4, 9, 7, 6, 13]
ws = [5, 6, 4, 3, 7]

V = [[0 for i in range(W + 1)] for j in range(n + 1)]

for i in range(1, n + 1):
    v_i = vs[i - 1]
    w_i = ws[i - 1]
    for w in range(1, W + 1):
        k = 1
        while True:
            if k*w_i > w:
                V[i][w] = max(V[i][w], V[i-1][w])
                break
            else:
                V[i][w] = max(V[i][w], k*v_i + V[i-1][w - k*w_i])
            k += 1
pprint.pprint(V)
```

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 4, 4, 4, 4, 4, 8, 8],
 [0, 0, 0, 0, 0, 4, 9, 9, 9, 9, 9, 13],
 [0, 0, 0, 0, 7, 7, 9, 9, 14, 14, 16, 16],
 [0, 0, 0, 6, 7, 7, 12, 13, 14, 18, 19, 20],
 [0, 0, 0, 6, 7, 7, 12, 13, 14, 18, 19, 20]]
```

In [7]:
```
# solution for 3.2

prob = LpProblem("Knapsack with Replacement",LpMaximize)
x1=LpVariable("x1", 0, None, LpInteger)
x2=LpVariable("x2", 0, None, LpInteger)
```

```python
        x3=LpVariable("x3", 0, None, LpInteger)
        x4=LpVariable("x4", 0, None, LpInteger)
        x5=LpVariable("x5", 0, None, LpInteger)

        prob += 4*x1 + 9*x2 + 7*x3 + 4*x4 + 13*x5
        prob += 5*x1 + 6*x2 + 4*x3 + 3*x4 + 7*x5 <= 11

        prob.solve()
        val = value(prob.objective)
        print("Optimal Value for Knapsack with Replacement")
        for v in prob.variables():
            print("{} = {}".format(v.name, v.varValue))
        print("z = {}".format(val))
```

```
Optimal Value for Knapsack with Replacement
x1 = 0.0
x2 = 0.0
x3 = 1.0
x4 = 0.0
x5 = 1.0
z = 20.0
```