In [1]:
```
# import pulp
from pulp import LpVariable, LpProblem, LpMaximize, LpStatus, value, LpMinimize
```

Question 1

SteelCo manufactures three types of steel at two different steel mills. During a given month, Mill 1 has 200 hours of blast furnace time available, whereas Mill 2 has 300 hours. Because of differences in the furnaces at each mill, the time and cost to produce a ton of steel differs for each mill and are shown in the following table. Each month, SteelCo must manufacture a total of at least 400 tons of Steel 1, 500 tons of Steel 2, and 300 tons of Steel 3 to meet demand; however, the total amount of Steel 2 manufactured should not exceed the combined amount of Steel 1 and Steel 3. Also, in order to maintain a roughly uniform usage of the two mills, management's policy is that the percentage of available blast furnace capacity (time) used at each mill should be the same. Clearly formulate a linear program (LP) to minimize the cost of manufacturing the desired steel.

In [2]:
```
# define variables

s1m1 = LpVariable("steel1Mill1", 0, None)
s1m2 = LpVariable("steel1Mill2", 0, None)
s2m1 = LpVariable("steel2Mill1", 0, None)
s2m2 = LpVariable("steel2Mill2", 0, None)
s3m1 = LpVariable("steel3Mill1", 0, None)
s3m2 = LpVariable("steel3Mill2", 0, None)
bfm1 = LpVariable("blastFurnaceMill1capacity", 0, 200)  # Mill 1 has 200 hours of blast furnace ti
bfm2 = LpVariable("blastFurnaceMill2capacity", 0, 300)  # Mill 2 has 300 hours
```

In [3]:
```
# defines the problem
prob = LpProblem("problem", LpMinimize)
```

In [4]:
```
# define constraints - comments on end provide context
prob += s1m1 + s1m2  == 400  # SteelCo must manufacture a total of at least 400 tons of Steel 1
prob += s2m1 + s2m2  == 500  # 500 tons of Steel 2
prob += s3m1 + s3m2  == 300  # 300 tons of Steel 3
prob += s2m1 + s2m2  <= s1m1 + s1m2  + s3m1 + s3m2  # total amount of Steel 2 manufactured should
prob += (s1m1 + s2m1 + s3m1)/200 == (s1m2 + s2m2 + s3m2)/300  # the percentage of available blast
prob += (20/60)*s1m1 + (22/60)*s2m1 + (28/60)*s3m1 <= 200  # values from table
prob += (24/60)*s1m2 + (18/60)*s2m2 + (30/60)*s3m2 <= 300  # values from table
```

In [5]:
```
# define objective function
prob += 10*s1m1 + 11*s2m1 + 14*s3m1 + 12*s1m2 + 9*s2m2 + 10*s3m2
```

In [6]:
```
# solve the problem
status = prob.solve()
print(f"Problem")
print(f"status={LpStatus[status]}")

# print the results
for variable in prob.variables():
    print(f"{variable.name} = {variable.varValue}")

print(f"Objective = {value(prob.objective)}")
print(f"")
```

```
Problem
status=Optimal
steel1Mill1 = 400.0
steel1Mill2 = 0.0
steel2Mill1 = 80.0
steel2Mill2 = 420.0
steel3Mill1 = 0.0
steel3Mill2 = 300.0
Objective = 11660.0
```

Steel mill 1 should produce 400 tons of steel1, 80 of steel2, and 0 of steel3 mill 2 should produce 0 tons of steel1, 420 of steel2 and all 300 tons of steel 3

this will result in an optimally minimized furnace time of 11,660hrs

Question 2

Consider the following linear program:

Max Z = -4x1 + 2x2
Subject To
(a) -2x1 + 2x2 ≤ 7
(b) x1 ≥ 2
(c) x1 - 4x2 ≤ 0
(d) 2x1 + 2x2 ≥ 10
(e) x1, x2 ≥ 0

Part A: Write the LP in standard equality form.

objective function is to be minimized, so we convert the objective: Z = 4 * x1 - 2 * x2# constraints should be lower bounds (a) is already lower -2x1 + 2x2 ≤ 7 (b) needs signs flipped: -x1 <= -2 (c) is good (d) needs signs flipped -2x1 - 2x2 <= -10 (e) is a positivity constraint for both variables, so they're fine & help with the positivity constraint requirement. x1 is also overcome by constraint (b) x2 ≥ 0 # Equalities should be spit into inequalities none exit, no work needed# all variables need positivity constraints, as mentioned above - already set

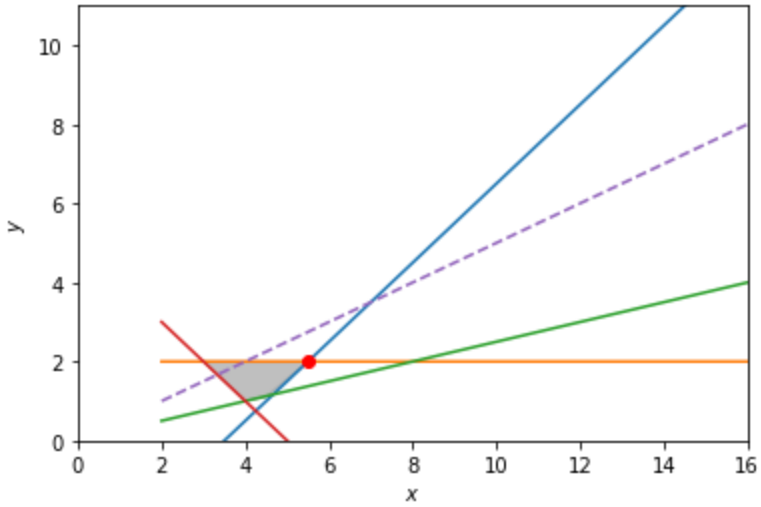
Part B: Solve the original LP graphically (to scale). Clearly identify the feasible region and, if one or more exist, the optimal solution(s) (provide exact values for x1, x2, and Z).

In [7]:

```
#### resource: https://benalexkeen.com/linear-programming-with-python-and-pulp-part-1/
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

# Construct lines
# (e)x2 >= 0   # need to define x first
x = np.linspace(2, 20, 2000)

# (a) -2x1 + 2x2 ≤ 7
y1 = x-3.5

# (b) x1 ≥ 2  - adjust definition from (e)
y2 = (x*0) + 2

# (c) x1 - 4x2 ≤ 0
y3 = .25*x

# (d) 2x1 + 2x2 ≥ 10
```

```
y4 = 5-x

# (e) second part - made irrelevant by (b)

# objective: Max Z = -4x1 + 2x2
y5 = .5 * x


# Make plot
plt.plot(x, y1, label=r'$2x1 + 2x2\leq7$')  # (a)
plt.plot(x, y2, label=r'$x1\geq2$') # (b)
plt.plot(x, y3, label=r'$x1 - 4x2\leq 0$')  # (c)
plt.plot(x, y4, label=r'$2x1 + 2x2\geq 10$')  # (d)
plt.plot(x, y5, label=r'$y\leq 2$', linestyle='dashed')  # (e)
plt.xlim((0, 16))
plt.ylim((0, 11))
plt.xlabel(r'$x$')
plt.ylabel(r'$y$')


idx = np.argwhere(np.diff(np.sign(y1 - y2))).flatten()
print('value of x:', x[idx], 'value of y:', y1[idx], 'value of z:', (-4*y1[idx] + 2*x[idx]))
# max in direction of objective is where y1 meets y2:
plt.plot(x[idx], y1[idx], 'ro')

# Fill feasible region - can't get this to leave out that bottom triangle....
# y6 = np.maximum(y1, y2)
# y7 = np.maximum(y4, y1)
# plt.fill_between(x, y6, y7, where=y6>y7, color='grey', alpha=0.5)
# plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

x = [3,4,4.5,5.5]
y = [2,1,1.1,2]

plt.fill(x, y, color='grey', alpha=0.5)
plt.show()
```

```
value of x: [5.49374687] value of y: [1.99374687] value of z: [3.01250625]
```



validate

In [8]:

```
# define variables

x1 = LpVariable("x1", 2, 50)
x2 = LpVariable("x2", 0, 30)
```

In [9]:

```
# defines the problem
prob2 = LpProblem("problem", LpMaximize)
```

In [10]:
```
# define constraints
prob2 += -2*x1 + 2*x2 <= 7
prob2 += x1 >= 2
prob2 += x1 - 4*x2 <= 0
prob2 += 2*x1 + 2*x2 >= 10
prob2 += x2 >= 0
```

In [11]:
```
# define objective function
prob2 += -4*x1 + 2*x2
```

In [14]:
```
# solve the problem
status2 = prob2.solve()
print(f"Problem")
print(f"status={LpStatus[status2]}")

# print the results
for variable in prob2.variables():
    print(f"{variable.name} = {variable.varValue}")

print(f"Objective = {value(prob2.objective)}")
print(f"")
```

```
Problem
status=Optimal
x1 = 2.0
x2 = 5.5
Objective = 3.0
```

Question 3

At the beginning of month 1, Finco has $400 in cash. At the beginning of months 1, 2, 3, and 4, Finco receives certain revenues, after which it pays bills (see Table 2 below). Any money left over may be invested for one month at the interest rate of 0.1% per month; for two months at 0.5% per month; for three months at 1% per month; or for four months at 2% per month. Use linear programming to determine an investment strategy that maximizes cash on hand at the beginning of month 5. Formulate an LP to maximize Finco's profit.

In [15]:
```
########### problem definitions

# 1 month `a` = 0.1% per month
# 2 month `b` = 0.5% per month
# 3 month `c` = 1.0% per month
# 4 month `d` = 2.0% per month
# amounts invested month 1: a1, b1, c1, d1
# amounts invested month 1: a2, b2, c2
# amounts invested month 1: a3, b3
# amounts invested month 1: a4
```

In [16]:
```
# define variables
a1 = LpVariable("inventmentAmonth1", 0, None)
b1 = LpVariable("inventmentBmonth1", 0, None)
c1 = LpVariable("inventmentCmonth1", 0, None)
d1 = LpVariable("inventmentDmonth1", 0, None)
```

```
a2 = LpVariable("inventmentAmonth2", 0, None)
b2 = LpVariable("inventmentBmonth2", 0, None)
c2 = LpVariable("inventmentCmonth2", 0, None)
a3 = LpVariable("inventmentAmonth3", 0, None)
b3 = LpVariable("inventmentBmonth3", 0, None)
a4 = LpVariable("inventmentAmonth4", 0, None)
```

In [17]:

```
# defines the problem
prob3 = LpProblem("problem", LpMaximize)
```

In [18]:

```
 # define constraints
prob3 += 400 + 400 >= 600 + a1 + b1 + c1 + d1 # year 1 in = out
prob3 += a1*(1.001) + 800 >= 500 + a2 + b2 + c2
prob3 += a2*(1.001) + b1*(1.005**2) + 300 >= 500 + a3 + b3
prob3 += a3*(1.001) + b2*(1.005**2) + c1*(1.01**3) + 300 >= 250 + a4
```

In [19]:

```
# define objective function
prob3 += a4*(1.001) + b3*(1.005**2) + c2*(1.01**3) + d1*(1.02**4)
```

In [22]:

```
# solve the problem
status3 = prob3.solve()
print(f"Problem")
print(f"status={LpStatus[status3]}")

# print the results
for variable in prob3.variables():
    print(f"{variable.name} = {variable.varValue}")

print(f"Objective = {value(prob3.objective)}")
print(f"")
```

```
Problem
status=Optimal
inventmentAmonth1 = 0.0
inventmentAmonth2 = 199.8002
inventmentAmonth3 = 0.0
inventmentAmonth4 = 50.0
inventmentBmonth1 = 0.0
inventmentBmonth2 = 0.0
inventmentBmonth3 = 0.0
inventmentCmonth1 = 0.0
inventmentCmonth2 = 100.1998
inventmentDmonth1 = 200.0
Objective = 369.7723861398
```

in month 1, all $200 remaining should be invested in (d) in month 2, approx \$100 should be invested in (c), and $200 in (a) in month 3, no additional money is invested, as the \$200 return from month 2 is used to pay bills in month 4, $50 should be invested in (a) this will result in approx \$370 cash on hand in month 5

Question 4

Turkeyco produces two types of turkey cutlets for sale to fast-food restaurants. Each type of cutlet consists of white meat and dark meat. Cutlet 1 sells for $4/lb and must consist of at least 70\% white meat. Cutlet 2 sells for \$3/lb and must consist of at least 60% white meat. At most, 50 lb of cutlet 1 and 30 lb of cutlet 2 can be sold.

The two types of turkey used to manufacture the cutlets are purchased from the GobbleGobble Turkey Farm. Each type 1 turkey costs $10 and yields 5 lb of white meat and 2 lb of dark meat. Each type 2 turkey costs \$8 and yields 3 lb of white meat and 3 lb of dark meat.

In [23]:
```
# define variables

c1 = LpVariable("cutlet1amount", 0, 50) # At most, 50 lb of cutlet 1 can be sold
c2 = LpVariable("cutlet2amount", 0, 30) # At most 30 lb of cutlet 2 can be sold
t1 = LpVariable("turkey1count", 0, None,cat='Integer')
t2 = LpVariable("turkey2count", 0, None,cat='Integer')
wm = LpVariable("lbsWhiteMeat", 0, None)
dm = LpVariable("lbsDarkMeat", 0, None)
```

In [24]:
```
# defines the problem
prob4 = LpProblem("problem", LpMaximize)
```

In [25]:
```
# define constraints - comments on end provide context
prob4 += 5*t1 + 3*t2 == wm  #  type 1 turkey yields 5 lb of white meat; 2: 3 lb
prob4 += 2*t1 + 3*t2 == dm  #  type 1 turkey yields 2 lb of dark meat; 2: 3 lb
prob4 += .7*c1 + .6*c2 <= wm   # Cutlet 1 must consist of at least 70% white meat, 2- 60% white me
```

In [26]:
```
# define objective function
prob4 += 4*c1 + 3*c2 - t1*10 - t2*8  # Cutlet 1 sells for $4/lb, 2 sells for $3/lb; type 1 turkey
```

In [27]:
```
# solve the problem
status4 = prob4.solve()
print(f"Problem")
print(f"status={LpStatus[status4]}")

# print the results
for variable in prob4.variables():
    print(f"{variable.name} = {variable.varValue}")

print(f"Objective = {value(prob4.objective)}")
print(f"")
```

```
Problem
status=Optimal
cutlet1amount = 50.0
cutlet2amount = 30.0
lbsDarkMeat = 23.0
lbsWhiteMeat = 53.0
turkey1count = 10.0
turkey2count = 1.0
Objective = 182.0
```

ten type 1 turkeys and one type 2 turkeys should be used to produce 50lbs of cutlet 1 and 30lbs and cutlet 2, resulting in maximized profit of $182

Question 5

A company wants to plan production for the ensuing year to minimize the combined cost of production and inventory costs. In each quarter of the year, demand is anticipated to be 130, 160, 250, and 150 units,

respectively. The plant can produce a maximum of 200 units each quarter. The product can be manufactured at a cost of $15 per unit during the first quarter, however the manufacturing cost is expected to rise by \$1 per quarter. Excess production can be stored from one quarter to the next at a cost of $1.50 per unit, but the storage facility can hold a maximum of 60 units. How should the production be scheduled so as to minimize the total costs?

In [28]:
```
# define variables

p1 = LpVariable("q1produced", 0, 200) # units produced q1
p2 = LpVariable("q2produced", 0, 200) # units produced q2
p3 = LpVariable("q3produced", 0, 200) # units produced q3
p4 = LpVariable("q4produced", 0, 200) # units produced q4
s1 = LpVariable("q1stored", 0, 60) # units stored q1
s2 = LpVariable("q2stored", 0, 60) # units stored q2
s3 = LpVariable("q3stored", 0, 60) # units stored q3
s4 = LpVariable("q4stored", 0, 60) # units stored q4
```

In [29]:
```
# defines the problem
prob5 = LpProblem("problem", LpMinimize)
```

In [30]:
```
# define constraints - comments on end provide context
prob5 += p1 >= 130 + s1
prob5 += p2 + s1 >= 160 + s2
prob5 += p3 + s2 >= 250 + s3
prob5 += p4 + s3 >= 150 + s4
```

In [31]:
```
# define objective function
prob5 += p1*15 + p2*16 + p3*17 + p4*18 + (s1 + s2 + s3 + s4)*1.5
```

In [32]:
```
# solve the problem
status5 = prob5.solve()
print(f"Problem")
print(f"status={LpStatus[status5]}")

# print the results
for variable in prob5.variables():
    print(f"{variable.name} = {variable.varValue}")

print(f"Objective = {value(prob5.objective)}")
print(f"")
```

```
Problem
status=Optimal
q1produced = 140.0
q1stored = 10.0
q2produced = 200.0
q2stored = 50.0
q3produced = 200.0
q3stored = 0.0
q4produced = 150.0
q4stored = 0.0
Objective = 11490.0
```

In quarter 1, 140 units should be produced with 10 stored in quarter 2, 200 units should be produced with 50 stored in quarter 3, 200 units should be produced with 0 stored (10 produced in q1 & 40 produced in q2 were saved to supply over capacity) in quarter 4, the demand should be met at 150 units

this will result in a minimized cost of $11,490