Robert Werthman
CSCI 5654
Homework 1

P1.

1. First change the problem into a maximization problem:

$$\text{minimize } 3x_1 - 5x_2 \Rightarrow \text{maximize } -3x_1 + 5x_2$$

2. Change the constraints to $\leq$:

$$4x_1 + x_2 \geq -4 \Rightarrow -4x_1 - x_2 \leq 4$$

$$2x_1 - x_2 \geq -8 \Rightarrow -2x_1 + x_2 \leq 8$$

3. Make sure the variables have non-negativity constraints:

$$x_2 \mapsto x_2^+ - x_2^-$$

$$x_2^+, x_2^- \geq 0$$

   Replace x_2 with $x_2^+ - x_2^-$ in the problem.

The problem can now be written in standard form as:

$$\begin{aligned} \textbf{maximize } & -3x_1 + 5x_2^+ - 5x_2^- \\ \textbf{s.t. } & -4x_1 - x_2^+ + x_2^- \leq 4 \\ & -2x_1 + x_2^+ - x_2^- \leq 8 \\ & x_1 + 2x_2^+ - 2x_2^- \leq 4 \\ & x_1, x_2^+, x_2^- \geq 0 \end{aligned}$$

P2.

Slack, initial dictionary, feasibility

I will add the slack variables w_1, w_2, and w_3 to the constraints and let z be the value of the objective function.
Rewriting the problem with the slack variables gives:

$$\begin{aligned}
\textbf{maximize } z &= -3x_1 + 5x_2^+ - 5x_2^- \\
\textbf{s.t. } w_1 &= 4 + 4x_1 + x_2^+ - x_2^- \\
w_2 &= 8 + 2x_1 - x_2^+ + x_2^- \\
w_3 &= 4 - x_1 - 2x_2^+ + 2x_2^- \\
& x_1, x_2^+, x_2^-, w_1, w_2, w_3 \geq 0
\end{aligned}$$

The first dictionary of this problem can be written as:

$$\begin{array}{l}
w_1 = 4 + 4x_1 + x_2^+ - x_2^- \\
w_2 = 8 + 2x_1 - x_2^+ + x_2^- \\
w_3 = 4 - x_1 - 2x_2^+ + 2x_2^- \\
\hline
z = 0 - 3x_1 + 5x_2^+ - 5x_2^-
\end{array}$$

A solution to this dictionary is feasible if and only if all the right-hand sides are nonnegative as set by the non-negativity constraints.

If we set the non-basic variables x_1, x_2^+, x_2^- to 0 then we have the solutions: $w_1 = 4$, $w_2 = 8$, $w_3 = 4$, and $z = 0$.

Since the non-negativity constraints are respected, this dictionary is feasible.

Initial pivoting

To pivot the dictionary we need to choose a variable from the objective funtion that if increased the value z of the objective function increases. This is the case because we are trying to maximize the value of the objective function. In this case, we will choose x_2^+ because it is the only variable that will increase the value of z. The other variables all of $-$ in front which means that if we increased those variables, we would decrease the value z of the objective function.

To find the leaving variable we need to solve for x_2^+ in all of the equations of the dictionary where their is a constraint on how much x_2^+can increase. We then choose the variable where x_2^+ has the lowest upper limit. We set all of the other variables to 0.

$$\begin{aligned}
& w_1 = 4 + 4x_1 + x_2^+ - x_2^- \Rightarrow \text{No constraint on the increase of } x_2^+ \\
& w_2 = 8 + 2x_1 - x_2^+ + x_2^- \Rightarrow x_2^+ \leq 8 \\
& w_3 = 4 - x_1 - 2x_2^+ + 2x_2^- \Rightarrow x_2^+ \leq 2
\end{aligned}$$

In this case, we choose w_3 as the leaving variable because it gives x_2^+ the lowest upper bound constraint.

Once we have the entering and leaving variables, we need to find the new dictionary after pivoting. To do this we first solve the equation of the leaving variable for the entering variable:

$$w_3 = 4 - x_1 - 2x_2^+ + 2x_2^- \Rightarrow x_2^+ = 2 - \frac{x_1}{2} + x_2^- - \frac{w_3}{2}$$

We then substitute that equation in for all places where the entering variable occurs in the dictionary:

$$\begin{aligned}
w_1 &= 4 + 4x_1 + (2 - \frac{x_1}{2} + x_2^- - \frac{w_3}{2}) - x_2^- \\
w_2 &= 8 + 2x_1 - (2 - \frac{x_1}{2} + x_2^- - \frac{w_3}{2}) + x_2^- \\
x_2^+ &= 2 - \frac{x_1}{2} + x_2^- - \frac{w_3}{2} \\
z &= 0 - 3x_1 + 5(2 - \frac{x_1}{2} + x_2^- - \frac{w_3}{2}) - 5x_2^-
\end{aligned}$$

After some algebra, the new dictionary after pivoting looks like:

$$\begin{array}{l}
w_1 = 6 + \frac{7}{2}x_1 - \frac{w_3}{2} \\
w_2 = 6 + \frac{3}{2}x_1 + \frac{w_3}{2} \\
x_2^+ = 2 - \frac{x_1}{2} + x_2^- - \frac{w_3}{2} \\
\hline
z = 10 - \frac{11}{2}x_1 - \frac{5}{2}w_3
\end{array}$$

P3.

Linear programming model of the problem

The client needs to invest money in each of investments options in order to maximize their profit. Let x_i be the amount of money invested in the investment option with ID i. Let epu_i be the expected profit per unit for the investment option with ID i. Let pu_i be the price per unit for the investment option with ID i. The objective function for this maximization problem can then be written as:

$$\textbf{maximize} \sum_{i=1}^{15} \frac{epu_i}{pu_i} x_i$$

This says that the total expected profit of all the investments is equal to the sum of the amount of money invested in each investment option divided by the price per unit for each investment option multiplied by the expected profit per unit for each investment option. This is subject to the constraints:

1. $\sum_{i=1}^{15} x_i \leq 10000$
   - Total investment is at most \$10,000.
2. $1500 \leq x_1 + x_4 + x_{10} + x_{13} \leq 3500$
   - The minimum and maximum investment in risk category A.
3. $4500 \leq x_2 + x_5 + x_8 + x_9 + x_{14} \leq 6500$
   - The minimum and maximum investement in risk category B.
4. $1000 \leq x_3 + x_7 + x_{11} + x_{15} \leq 3000$ - The minimum and maximum investement in risk category C.
5. $500 \leq x_6 + x_{12} \leq 2500$
   - The minimum and maximum investement in risk category D.
6. $0 \leq x_1 + x_8 + x_{11} \leq 3000$
   - The minimum and maximum investement in investement market Tech.
7. $0 \leq x_2 + x_3 + x_5 + x_6 + x_7 + x_{15} \leq 4000$
   - The minimum and maximum investement in investement market Finance.
8. $0 \leq x_4 + x_9 + x_{13} \leq 5000$
   - The minimum and maximum investement in investement market PetroChem.
9. $0 \leq x_{10} + x_{12} + x_{14} \leq 7000$
   - The minimum and maximum investement in investement market Automobile.
10. $2000 \leq x_1 + x_2 + x_3 + x_6 + x_7 + x_8 + x_9 + x_{10} + x_{11} + x_{13} \leq 10000$
    - The minimum and maximum investement in EcoFriendly.
11. $x_1, ..., x_{15} \geq 0$
    - You cannot invest a negative amount of money in an investment option.

Solution

I used the PyGLPK python library to solve this problem. The problem turned out to be feasible with none of the constraints contradicting each other. The problem was also unbounded because the maximum of the objective functions was not ∞ or $-\infty$.

- The optimal value z of the objective function to maximize profit is \$8513.47.

- $3000 dollars should be invested in the investment option with ID 3 (x_3).
- $1500 dollars should be invested in the investment option with ID 10 (x_{10}).
- $500 dollars should be invested in the investment option with ID 12 (x_{12}).
- $5000 dollars should be invested in the investment option with ID 14 (x_{14}).
- Nothing should be invested in the other investment options.

P4.

(A)

Let two solutions $y, z \in F$ and let the solution $x = \lambda y + (1-\lambda)z$ which means $x \in F$. Then we can say

$$Ax = A(\lambda y + (1-\lambda)z) = \lambda Ay + (1-\lambda)Az \leq \lambda b + (1-\lambda)b = b$$

The set of feasible solutions F is convex, because any solution x made up of two feasible solutions $y, z \in F$ and of the form $\lambda y + (1-\lambda)z$ respects the equation $Ax \leq b$. This makes x a feasible solution $\in F$ and the set F convex.

(B)

Both optimal solutions are in a convex set of $O : \{x | c^t x = z^*\}$. If there are infinitely many optimal solutions in O then there exists a point equal to $\lambda x_1 + (1-\lambda)x_2$ along the line connecting x_1 and x_2 that is also equal z^*.

Let $y = x_1$ and $w = x_2$ then the optimal objective value is

$$z^* = C^t y = C^t w$$

Let $x = \lambda y + (1-\lambda)w$ then we can say

$$C^t x = C^t(\lambda y + (1-\lambda)w)) = \lambda C^t y + (1-\lambda)C^t w = \lambda z^*(1-\lambda)z^* = z^*$$

It was shown that the solution x of the form $\lambda y + (1-\lambda)w$ equals the optimal objective value z^*. This means x is in the set O and that the set O is convex. This means there are infinitely many optimal solutions of the form $\lambda y + (1-\lambda)w$ where $\lambda \in [0, 1]$ in the optimal solution set O.

P5.

(A)

Linear programming model of the problem

The goal of this problem is to minimize the total cost of food while meeting all of the caloric nutrient needs. Let x_i represent the food item i. Let ps_i represent the price per serving of food item i. The objective function of this minimization problem can then be written as:

$$\textbf{minimize } .5x_0 + 2.5x_1 + .25x_2 + .2x_3 + .6x_4$$

This says that to minimize the cost of the food that is purchased we need to choose the optimal quantities of each food item. This is subject to the constraints:

1. $1800 \leq 300x_0 + 550x_1 + 450x_2 + 25x_3 + 300x_4 \leq 2200$
   - The minimum and maximum calories consumed for all of the food items.
2. $50 \leq 20x_0 + 25x_1 + 25x_2 + 4x_3 + 15x_4 \leq 100$
   - The minimum and maximum carbs consumed for all of the food items.
3. $30 \leq 5x_0 + 8x_1 + 4x_2 + 2x_3 + 3x_4 \leq 80$
   - The minimum and maximum protein consumed for all of the food items.
4. $60 \leq 10x_0 + 20x_1 + 5x_2 + .5x_3 + .5x_4 \leq 100$
   - The minimum and maximum fats consumed for all of the food items.
5. $3 \leq .1x_0 + .9x_1 + .1x_2 + .1x_3 + .1x_4 \leq 5$
   - The minimum and maximum sodium consumed for all of the food items.
6. $x_0, x_1, x_2, x_3, x_4 \geq 0$
   - The amount of each nutrient consumed cannot be negative.

Solution

I used the PyGLPK python library to solve this problem. It used the Simplex algorithm to solve Linear Programming problems. It gave the following answers:

- The optimal value z of the objective function to minimize cost is \$7.96.
- 2.76 servings of Ramen x_1 should be purchased.
- .49 servings of Rice x_2 should be purchased.
- 4.67 servings of Brocolli x_3 should be purchased.
- No servings of cookies x_0 or cornflakes x_4 should be purchased.

(B)

The constraints below should be added to ensure that no single food accounts for more than 50% of the total caloric intake:

1. $300x_0 \leq 0.5(300x_0 + 550x_1 + 450x_2 + 25x_3 + 300x_4)$
   - The calories from Cookies should not account for more than 50% of the total caloric intake.
2. $550x_1 \leq 0.5(300x_0 + 550x_1 + 450x_2 + 25x_3 + 300x_4)$
   - The calories from Ramen should not account for more than 50% of the total caloric intake.

3. $450x_2 \leq 0.5(300x_0 + 550x_1 + 450x_2 + 25x_3 + 300x_4)$
   - The calories from Rice should not account for more than 50% of the total caloric intake.
4. $25x_3 \leq 0.5(300x_0 + 550x_1 + 450x_2 + 25x_3 + 300x_4)$
   - The calories from Broccoli should not account for more than 50% of the total caloric intake.
5. $300x_4 \leq 0.5(300x_0 + 550x_1 + 450x_2 + 25x_3 + 300x_4)$
   - The calories from CornFlakes should not account for more than 50% of the total caloric intake.

Solution

I used the command line tool **glpsol** that comes with GLPK to solve the problem. I used the GNU MathProg modeling language (GMPL) to create a .mod file with the linear programming model of the problem. The solver said that with these new constraints there was no feasible solution to this problem.

(C)

```
"""
Robert Werthman
CSCI 5654
Homework 1 Problem 3

maximize (expected profit per unit)/(price per unit) x1 +...+
         (expected profit per unit)/(price per unit) x15

subject to:
  0 <= x1 +...+ x15 <= 10000
  1500 <= x1 + x4 + x10 + x13 <= 3500
  4500 <= x2 + x5 + x8 + x9 + x14 <= 6500
  1000 <= x3 + x7 + x11 + x15 <= 3000
  500 <= x6 + x12 <= 2500
  0 <= x1 + x8 + x11 <= 3000
  0 <= x2 + x3 + x5 + x6 + x7 + x15 <= 4000
  0 <= x4 + x9 + x13 <= 5000
  0 <= x10 + x12 + x14 <= 7000
  2000 <= x1 + x2 + x3 + x6 + x7 + x8 + x9 + x10 + x11 + x13 <= 10000
  x1,...,x15 >= 0
"""
import glpk

lp = glpk.LPX()
lp.name = 'p3' # Assign a name to the problem.
lp.obj.maximize = True # Treat this as a maximization problem.

lp.rows.add(10) # This is the number of constraint equations
for r in lp.rows:
  r.name = chr(ord('a') + r.index) # Names rows starting with a through j.
  print r.name

# Set the bounds for the constraint equations.
lp.rows[0].bounds = 0.0, 10000.0
lp.rows[1].bounds = 1500.0, 3500.0
lp.rows[2].bounds = 4500.0, 6500.0
lp.rows[3].bounds = 1000.0, 3000.0
lp.rows[4].bounds = 500.0, 2500.0
lp.rows[5].bounds = 0.0, 3000.0
lp.rows[6].bounds = 0.0, 4000.0
lp.rows[7].bounds = 0.0, 5000.0
lp.rows[8].bounds = 0.0, 7000.0
lp.rows[9].bounds = 2000.0, 10000.0

lp.cols.add(15) # The total number of variables in the problem.

for c in lp.cols:
  c.name = 'x%d' % (c.index + 1) # Name the variables starting with x1.
  c.bounds = 0.0, None # Set the bounds of the variables to be >= 0

# Set the objective function coefficients.
lp.obj[:] = [1.451/2.563, 2.683/4.307, 5.898/6.422, 2.102/3.488, 5.709/6.581,
             4.519/8.993, 7.176/11.481, 6.075/11.730, 5.718/9.270, 7.442/10.160,
```

```
            1.234/1.961, 4.680/9.300, 7.229/11.672, 9.589/10.877, 6.497/12.137]

# Set the coefficients for each of the variables in the constraint equations.
lp.matrix = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
             0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 1, 0,
             0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1,
             0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0,
             1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0,
             0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1,
             0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0,
             0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0,
             1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0]

lp.simplex() # Solve this LP with the simplex method
print 'z = %g;' % lp.obj.value # Retrieve and print objective function value.

# Print the other variables names and values.
print '; '.join('%s = %g' % (c.name, c.primal) for c in lp.cols)

"""
Robert Werthman
CSCI 5654
Homework 1: Problem 5
"""
import glpk

def part_a():
  """
  Homework 1 Problem 5a
  minimize .5x0 + 2.5x1 + .25x2 + .2x3 + .6x4

  subject to:
    1800 <= 300x0 + 550x1 + 450x2 + 25x3 + 300x4 <= 2200
    50 <= 20x0 + 25x1 + 25x2 + 4x3 + 15x4 <= 100
    30 <= 5x0 + 8x1 + 4x2 + 2x3 + 3x4 <= 80
    60 <= 10x0 + 20x1 + 5x2 + .5x3 + .5x4 <= 100
    3 <= .1x0 + .9x1 + .1x2 + .1x3 + .1x4 <= 5
    x0,x1,x2,x3,x4 >= 0
  """

  lp = glpk.LPX()
  lp.name = 'p5' # Assign a name to the problem.
  lp.obj.maximize = False # Treat this as a maximization problem.

  lp.rows.add(5) # This is the number of constraint equations.
  for r in lp.rows:
    r.name = chr(ord('a') + r.index) # Give names to the rows.
    print r.name

  # Set the bounds for the constraint equations.
  lp.rows[0].bounds = 1800.0, 2200.0
  lp.rows[1].bounds = 50.0, 100.0
```

```
  lp.rows[2].bounds = 30.0, 80.0
  lp.rows[3].bounds = 60.0, 100.0
  lp.rows[4].bounds = 3.0, 5.0

  lp.cols.add(5) # The total number of variables in the problem.
  for c in lp.cols:
    c.name = 'x%d' % c.index # Name the variables starting with x0.
    c.bounds = 0.0, None # Set the bounds of the variables to be >= 0

  # Set the objective function coefficients.
  lp.obj[:] = [0.5, 2.5, 0.25, 0.2, 0.6]

  # Set the coefficients for each of the variables in the constraint equations.
  lp.matrix = [300, 550, 450,  25, 300,
                20,  25,  25,   4,  15,
                 5,   8,   4,   2,   3,
                10,  20,   5, 0.5, 0.5,
               0.1, 0.9, 0.1, 0.1, 0.1]

  lp.simplex() # Solve this LP with the simplex method
  print 'z = %g;' % lp.obj.value # Retrieve and print objective function value.

  # Print the other variables names and values.
  print '; '.join('%s = %g' % (c.name, c.primal) for c in lp.cols)

  print lp.status # The solution status for the solver: infeas, unbnd

part_a()

/*
Robert Werthman
CSCI 5654
Homework 1: Problem 5 Part B

*/
# Define Variables
var x0 >= 0; # Cookie
var x1 >= 0; # Ramen
var x2 >= 0; # Rice
var x3 >= 0; # Brocolli
var x4 >= 0; # CornFlakes

# Define the objective function
minimize z: .5*x0 + 2.5*x1 + 0.25*x2 + 0.2*x3 + 0.6*x4;

# Define the constraints for 5A
s.t. a: 300*x0 + 550*x1 + 450*x2 + 25*x3 + 300*x4, <= 2200;
s.t. b: 300*x0 + 550*x1 + 450*x2 + 25*x3 + 300*x4, >= 1800;
s.t. c: 20*x0 + 25*x1 + 25*x2 + 4*x3 + 15*x4, <= 100;
s.t. d: 20*x0 + 25*x1 + 25*x2 + 4*x3 + 15*x4, >= 50;
s.t. e: 5*x0 + 8*x1 + 4*x2 + 2*x3 + 3*x4, <= 80;
s.t. f: 5*x0 + 8*x1 + 4*x2 + 2*x3 + 3*x4, >= 30;
s.t. g: 10*x0 + 20*x1 + 5*x2 + 0.5*x3 + 0.5*x4, <= 100;
```

```
s.t. h: 10*x0 + 20*x1 + 5*x2 + 0.5*x3 + 0.5*x4, >= 60;
s.t. i: 0.1*x0 + 0.9*x1 + 0.1*x2 + 0.1*x3 + 0.1*x4, <= 5;
s.t. j: 0.1*x0 + 0.9*x1 + 0.1*x2 + 0.1*x3 + 0.1*x4, >= 3;

# Define the constraints for 5B
s.t. k: 0.5*(300*x0 + 550*x1 + 450*x2 + 25*x3 + 300*x4), >= 300*x0;
s.t. l: 0.5*(300*x0 + 550*x1 + 450*x2 + 25*x3 + 300*x4), >= 550*x1;
s.t. m: 0.5*(300*x0 + 550*x1 + 450*x2 + 25*x3 + 300*x4), >= 450*x2;
s.t. n: 0.5*(300*x0 + 550*x1 + 450*x2 + 25*x3 + 300*x4), >= 25*x3;
s.t. o: 0.5*(300*x0 + 550*x1 + 450*x2 + 25*x3 + 300*x4), >= 300*x4;

# Solve
solve;
end;
```

```
Problem:    p5
Rows:       16
Columns:    5
Non-zeros:  80
Status:     UNDEFINED
Objective:  z = 0 (MINimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 z            B              0
     2 a            B              0                        2200
     3 b            B              0          1800
     4 c            B              0                         100
     5 d            B              0            50
     6 e            B              0                          80
     7 f            B              0            30
     8 g            B              0                         100
     9 h            B              0            60
    10 i            B              0                           5
    11 j            B              0             3
    12 k            B              0            -0
    13 l            B              0            -0
    14 m            B              0            -0
    15 n            B              0            -0
    16 o            B              0            -0

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------- ------------- ------------- -------------
     1 x0           NL             0             0                       < eps
     2 x1           NL             0             0                       < eps
     3 x2           NL             0             0                       < eps
     4 x3           NL             0             0                       < eps
     5 x4           NL             0             0                       < eps

Karush-Kuhn-Tucker optimality conditions:

KKT.PE: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
```

```
         High quality

KKT.PB: max.abs.err = 1.80e+03 on row 3
        max.rel.err = 9.99e-01 on row 3
        PRIMAL SOLUTION IS INFEASIBLE

KKT.DE: max.abs.err = 2.50e+00 on column 2
        max.rel.err = 7.14e-01 on column 2
        DUAL SOLUTION IS WRONG

KKT.DB: max.abs.err = 0.00e+00 on row 0
        max.rel.err = 0.00e+00 on row 0
         High quality

End of output
```