ISYE 6501 Homework 11

In the videos, we saw the “diet problem”. (The diet problem is one of the first large-scale optimization problems to be studied in practice. Back in the 1930’s and 40’s, the Army wanted to meet the nutritional requirements of its soldiers while minimizing the cost.) In this homework you get to solve a diet problem with real data. The data is given in the file diet.xls.

1. Formulate an optimization model (a linear program) to find the cheapest diet that satisfies the maximum and minimum daily nutrition constraints, and solve it using PuLP. Turn in your code and the solution. (The optimal solution should be a diet of air-popped popcorn, poached eggs, oranges, raw iceberg lettuce, raw celery, and frozen broccoli. UGH!)

```
from pulp import *
import pandas as pd

data = pd.read_excel("D:/…/diet.xls", sheet_name = "Sheet1") # read all data

dataTable = data[0:64] # rows 0-64
dataTable = dataTable.values.tolist() # convert to list

nutrientNames = list(data.columns.values) # column headers

minVal = data[65:66].values.tolist() # minimum
maxVal = data[66:67].values.tolist() # maximum

foods = [j[0] for j in dataTable] #list of food names

cost = dict([(j[0], float(j[1])) for j in dataTable]) # cost for each food

nutrients = []
for i in range(0,11): # for loop running through each nutrient: 11 times starting with 0
    nutrients.append(dict([(j[0], float(j[i+3])) for j in dataTable])) # amount of
nutrient i in food j

prob = LpProblem('Food optimization', LpMinimize) # 2 parameters: "name" and "sense"

foodVars = LpVariable.dicts("Foods", foods, 0)

prob += lpSum([cost[f] * foodVars[f] for f in foods]), 'Total Cost'

#constraints
for i in range(0,11): # for loop running through each nutrient: 11 times starting with 0
    prob += lpSum([nutrients[i][j] * foodVars[j] for j in foods]) >= minVal[0][i+3], 'min
nutrient ' + nutrientNames[i]
    prob += lpSum([nutrients[i][j] * foodVars[j] for j in foods]) <= maxVal[0][i+3], 'max
nutrient ' + nutrientNames[i]

#optimization
```

```
prob.solve()

#print output
print()
print("---------The solution to the diet problem is----------")
for var in prob.variables():
    if var.varValue > 0:
        print(str(var.varValue)+" units of "+str(var).replace('Foods_','') )
print()
print("Total cost of food = $%.2f" % value(prob.objective))
```

---------The solution to the diet problem is----------
52.64371 units of Celery,_Raw
0.25960653 units of Frozen_Broccoli
63.988506 units of Lettuce,Iceberg,Raw
2.2929389 units of Oranges
0.14184397 units of Poached_Eggs
13.869322 units of Popcorn,Air_Popped

Total cost of food = $4.34

2. Please add to your model the following constraints (which might require adding more variables) and solve the new model:
a. If a food is selected, then a minimum of 1/10 serving must be chosen. (Hint: now you will need two variables for each food i: whether it is chosen, and how much is part of the diet.

You'll also need to write a constraint to link them.)

```
# Additional Constraint

# Constraint A
# If a food is eaten, must eat at least 0.1 serving
for food in foods:
    prob += foodVars[food] >= 0.1 * foodVars_selected[food]

# If any of a food is eaten, its binary variable must be 1
for food in foods:
    prob += foodVars_selected[food] >= foodVars[food]*0.0000001
```

---------The solution to the diet problem is----------
52.64371 units of Celery,_Raw
0.25960653 units of Frozen_Broccoli
63.988506 units of Lettuce,Iceberg,Raw
2.2929389 units of Oranges
0.14184397 units of Poached_Eggs
13.869322 units of Popcorn,Air_Popped

Total cost of food = $4.34
From this output, it looks like the constraint did not affect the solution at all.

b. Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected.

```
# Constraint B
# Include at most 1 of celery and frozen brocolli
prob += foodVars_selected['Frozen Broccoli'] + foodVars_selected['Celery, Raw'] <= 1
```

---------The solution to the diet problem is----------
43.154119 units of Celery,_Raw
80.919121 units of Lettuce,Iceberg,Raw
3.0765161 units of Oranges
2.0464575 units of Peanut_Butter
0.14184397 units of Poached_Eggs
13.181772 units of Popcorn,Air_Popped

Total cost of food = $4.49
The constraint removed the broccoli, lowered the celery, and increased the lettuce and oranges. It also added PB

c. To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected. [If something is ambiguous (e.g., should bean-and-bacon soup be considered meat?), just call it whatever you think is appropriate – I want you to learn how to write this type of constraint, but I don't really care whether we agree on how to classify foods!]

```
# Constraint C
# At least 3 kinds of meat/poultry/fish/eggs
prob += foodVars_selected['Roasted Chicken'] + foodVars_selected['Poached Eggs'] \
        + foodVars_selected['Scrambled Eggs'] + foodVars_selected['Bologna,Turkey'] \
        + foodVars_selected['Frankfurter, Beef'] +
foodVars_selected['Ham,Sliced,Extralean'] \
        + foodVars_selected['Kielbasa,Prk'] + foodVars_selected['Pizza W/Pepperoni'] \
        + foodVars_selected['Hamburger W/Toppings'] \
        + foodVars_selected['Hotdog, Plain'] + foodVars_selected['Pork'] \
        + foodVars_selected['Sardines in Oil'] + foodVars_selected['White Tuna in Water']
\
        + foodVars_selected['Chicknoodl Soup'] + foodVars_selected['Splt Pea&Hamsoup'] \
        + foodVars_selected['Vegetbeef Soup'] + foodVars_selected['Neweng Clamchwd'] \
        + foodVars_selected['New E Clamchwd,W/Mlk'] + foodVars_selected['Beanbacn
Soup,W/Watr'] >= 3
```

---------The solution to the diet problem is----------
42.399358 units of Celery,_Raw
0.1 units of Kielbasa,Prk
82.802586 units of Lettuce,Iceberg,Raw
3.0771841 units of Oranges
1.9429716 units of Peanut_Butter
0.1 units of Poached_Eggs
13.223294 units of Popcorn,Air_Popped
0.1 units of Scrambled_Eggs

Total cost of food = $4.51