Question 15.2

Question 15.2.1

Answer:

First, read the data. We need to split the data in to food data and nutrition constraints.

In [1]:
```
import pandas as pd
from IPython.display import display

data = pd.read_excel("C:\Data\week 7 data-summer\data 15.2\diet.xls", sheet_name = "Sheet1")

#Sperate the raw data in to food and maximum/minimum nutrition constraints
food = data[0:64].copy()
constraints = data[65:67].copy()  \
                         .iloc[:,2:].copy()  \
                         .rename(columns = {"Serving Size":"Constraints"})  \
                         .reset_index(drop=True)
```

In [2]:
```
food.head()
```

Out[2]:

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carbohydrates g	Dietary_Fiber g	Protein g	Vit_A IU	Vit_C IU	Calcium mg	Iron mg
0	Frozen Broccoli	0.16	10 Oz Pkg	73.8	0.0	0.8	68.2	13.6	8.5	8.0	5867.4	160.2	159.0	2.3
1	Carrots,Raw	0.07	1/2 Cup Shredded	23.7	0.0	0.1	19.2	5.6	1.6	0.6	15471.0	5.1	14.9	0.3
2	Celery, Raw	0.04	1 Stalk	6.4	0.0	0.1	34.8	1.5	0.7	0.3	53.6	2.8	16.0	0.2
3	Frozen Corn	0.18	1/2 Cup	72.2	0.0	0.6	2.5	17.1	2.0	2.5	106.6	5.2	3.3	0.3
4	Lettuce,Iceberg,Raw	0.02	1 Leaf	2.6	0.0	0.0	1.8	0.4	0.3	0.2	66.0	0.8	3.8	0.1

In [3]:
```
constraints
```

Out[3]:

	Constraints	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carbohydrates g	Dietary_Fiber g	Protein g	Vit_A IU	Vit_C IU	Calcium mg	Iron mg
0	Minimum daily intake	1500.0	30.0	20.0	800.0	130.0	125.0	60.0	1000.0	400.0	700.0	10.0
1	Maximum daily intake	2500.0	240.0	70.0	2000.0	450.0	250.0	100.0	10000.0	5000.0	1500.0	40.0

Next, we need to create a linear programming problem in PuLP. Since we want to minimize the total cost, parameter "sense" will be set as LpMinimize.

We need to create a list of the foods and set up the variables. In this problem, we want to figure how many servings of each type of food we want to add to the diet. (let's call this variable x)

We will set up an dictionary which will contain the variable (desired number of servings) for each type of food. The lower bound of the variables will be set as 0.

In [4]:
```
from pulp import *

prob = LpProblem("Diet Problem", LpMinimize)
```

```python
#Create list of foods
food_list = list(food["Foods"])

#set up the dictionary of variables x
x = LpVariable.dicts("Quantity", indices = food_list, lowBound = 0)
```

```
C:\Users\suiwe\AppData\Roaming\Python\Python39\site-packages\pulp\pulp.py:1352: UserWarning: Spaces are not permitted in the name. Converted to '_'
  warnings.warn("Spaces are not permitted in the name. Converted to '_'")
```

In the newxt step, we need to add the objective functions and the constraints to the model. Then we run the modle and solve for the optimal solution. The optimal solution is printed as below.

In [5]:
```python
# Add objective: Minimize total cost of the diet
prob += (lpSum([list(food[food["Foods"]==i]["Price/ Serving"])[0] * x[i] for i in food_list]),
         "Total Cost")

# Add constraints to the model, so that the maximum and minimum nutrition requirements are satisfied
for col in range(3, len(food.columns)):
    column = food.columns[col]

    min_requirement = constraints[column][0]
    max_requirement = constraints[column][1]

    prob += (lpSum([list(food[food["Foods"]==i][column])[0] * x[i] for i in food_list]) >= min_requirement,
             "{} Min Requirement".format(column))
    prob += (lpSum([list(food[food["Foods"]==i][column])[0] * x[i] for i in food_list]) <= max_requirement,
            "{} Max Requirement".format(column))
```

In [6]:
```python
#Solve the problem
prob.solve()

print(f"status: {prob.status}, {LpStatus[prob.status]}")
print(f"objective: {prob.objective.value()}")

# Print the number of servings for each food (>0) in the optimal solution
for var in prob.variables():
    if var.value() != 0:
        print(f"{var.name}: {var.value()}")
```

```
status: 1, Optimal
objective: 4.337116797399999
Quantity_Celery,_Raw: 52.64371
Quantity_Frozen_Broccoli: 0.25960653
Quantity_Lettuce,Iceberg,Raw: 63.988506
Quantity_Oranges: 2.2929389
Quantity_Poached_Eggs: 0.14184397
Quantity_Popcorn,Air_Popped: 13.869322
```

Question 15.2

Question 15.2.2

Answer:

In this part, we are going to add more constraints to the model. We will recreate the same model as last question. We are also going to introduce a binary variable y for each food. If y=1, it means a food is selected into our diet. If a food is not selected, y would equal to zero.

In [7]:
```
#create a new model
prob2 = LpProblem("Diet Problem Part 2", LpMinimize)

# Add objective: Minimize total cost of the diet
prob2 += (lpSum([list(food[food["Foods"]==i]["Price/ Serving"])[0] * x[i] for i in food_list]),
         "Total Cost")

#set up the dictionary of variables y
y = LpVariable.dicts("Selected", indices = food_list, cat = "Binary", lowBound = 0)
```

The Constraints of the model will be mostly the same as last question.

Let's first load the same constraints as last question into the model.

In [8]:
```
# Add constraints to the model, so that the maximum and minimum nutrition requirements are satisfied
for col in range(3, len(food.columns)):
    column = food.columns[col]

    min_requirement = constraints[column][0]
    max_requirement = constraints[column][1]

    prob2 += (lpSum([list(food[food["Foods"]==i][column])[0] * x[i] for i in food_list]) >= min_requirement,
             "{} Min Requirement".format(column))
    prob2 += (lpSum([list(food[food["Foods"]==i][column])[0] * x[i] for i in food_list]) <= max_requirement,
            "{} Max Requirement".format(column))
```

To make sure at least the number of serving is equal or larger than 1/10 for each type of food choosen, two new constraint is needed:

```
1. for each type of food choosen, x >= 0.1 *y
```

```
2. for each type of food choosen, x <= 100,000,000 * y (make sure the fodd is not selected when y = 0)
```

In [9]:
```
# number of serving >= 0.1 for each type of choosen food
for i in food_list:
    prob2 += (x[i] >= y[i] * 0.1, "{} at least 0.1 serving Requirement".format(i))
    prob2 += (x[i] <= y[i] * 100000000, "{} is selected Requirement".format(i))
```

Then, we add another constraint to make sure celery and frozen broccoli are not selected at the same time.

In [10]:
```
prob2 += (y["Celery, Raw"] + y['Frozen Broccoli'] <= 1, "Celery and Broccoli Requirement")
```

Last, we need to make sure at least 3 kinds of meat/poultry/fish/eggs are selected in the diet. Here are the meat/poultry/fish/eggs we can find in the food list:

Roasted Chicken; Poached Eggs; Scrambled Eggs; Bologna,Turkey; Frankfurter, Beef; Ham,Sliced,Extralean; Kielbasa,Prk; Hamburger W/Toppings; Hotdog, Plain; Pork; Sardines in Oil; White Tuna in Water

In [11]:
```
meat_list = ["Roasted Chicken", "Poached Eggs", "Scrambled Eggs",\
             "Bologna,Turkey", "Frankfurter, Beef", "Ham,Sliced,Extralean",\
             "Kielbasa,Prk", "Hamburger W/Toppings", "Hotdog, Plain", "Pork", "Sardines in Oil", "White Tuna in Water"]
prob2 += (lpSum(y[i] for i in meat_list) >= 3, "Meat Requirement")
```

Here's the optimal solution we find and the number of servings of each type of the food selected.

In [12]:

```
#Solve the problem
prob2.solve()

print(f"status: {prob2.status}, {LpStatus[prob2.status]}")
print(f"objective: {prob2.objective.value()}")

# Print the number of servings for each food (>0) in the optimal solution
for var in prob2.variables():
    if var.value() != 0:
        print(f"{var.name}: {var.value()}")
```

```
status: 1, Optimal
objective: 4.512543427000001
Quantity_Celery,_Raw: 42.399358
Quantity_Kielbasa,Prk: 0.1
Quantity_Lettuce,Iceberg,Raw: 82.802586
Quantity_Oranges: 3.0771841
Quantity_Peanut_Butter: 1.9429716
Quantity_Poached_Eggs: 0.1
Quantity_Popcorn,Air_Popped: 13.223294
Quantity_Scrambled_Eggs: 0.1
Selected_Celery,_Raw: 1.0
Selected_Kielbasa,Prk: 1.0
Selected_Lettuce,Iceberg,Raw: 1.0
Selected_Oranges: 1.0
Selected_Peanut_Butter: 1.0
Selected_Poached_Eggs: 1.0
Selected_Popcorn,Air_Popped: 1.0
Selected_Scrambled_Eggs: 1.0
```