Question 15.2

Part 1

For part 1 of this question, I manually create a list of every foods and their calories, nutrients content.

I then create a foodVars variable, which stand for the amount of food for each type.

I set up objective function as minimizing the total cost

```
# -*- coding: utf-8 -*-
# =============================================================================
# Set up
# =============================================================================
#Load pulp library for reference
from pulp import *
#load panda module for loading and data manipulation
import pandas as pd

# Load diet data set into Spyder
data = pd.read_excel("C:\\Users\\nqviet94\\Downloads\\dietSummer2018.xls")
# remove the last three row which are diet requirements in the spreadsheet
data = data[0:64]

#convert to list
data = data.values.tolist()

# list of food names
foods = [x[0] for x in data]
#calories for each food
calories = dict([(x[0], float(x[3])) for x in data])
#al other nutrients information for each food
totalFat = dict([(x[0], float(x[5])) for x in data])
cholesterol= dict([(x[0], float(x[4])) for x in data])
sodium= dict([(x[0], float(x[6])) for x in data])
carbohydrates= dict([(x[0], float(x[7])) for x in data])
dietaryFiber= dict([(x[0], float(x[8])) for x in data])
Protein= dict([(x[0], float(x[9])) for x in data])
VitA= dict([(x[0], float(x[10])) for x in data])
VitC= dict([(x[0], float(x[11])) for x in data])
Calcium= dict([(x[0], float(x[12])) for x in data])
Iron= dict([(x[0], float(x[13])) for x in data])

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


# =============================================================================
# Create a list of variables for optimization problem
# =============================================================================
problem1 = LpProblem('PuLPTutorial', LpMinimize)
#This decision variable represents how much food we eat/consume
foodVars = LpVariable.dicts("Foods", foods, 0)
#This decision variable represents whether we eat a certain food or not
chosenVars = LpVariable.dicts("Chosen", foods,0,1, 'Binary')

# =============================================================================
# Create Objective function
# =============================================================================

problem1 += lpSum([cost[f] * foodVars[f] for f in foods])
```

For constraints in part 1, I manually add constraints for upper and lower bound for calories and all nutrients.

```
# ================================================================================
# Create constraints for the optimization problem
# ================================================================================

#Basic upper and lower bounds on calories and all other nutrients that are required in our diet
problem1 += lpSum([calories[f] * foodVars[f] for f in foods]) >= 1500, 'min Calories'
problem1 += lpSum([calories[f] * foodVars[f] for f in foods]) <= 2500, 'max Calories'

problem1 += lpSum([totalFat[f] * foodVars[f] for f in foods]) >= 20, 'min Fat'
problem1 += lpSum([totalFat[f] * foodVars[f] for f in foods]) <= 70, 'max Fat'

problem1 += lpSum([cholesterol[f] * foodVars[f] for f in foods]) >= 30, 'min Cholesterol'
problem1 += lpSum([cholesterol[f] * foodVars[f] for f in foods]) <= 240, 'max Cholesterol'

problem1 += lpSum([sodium[f] * foodVars[f] for f in foods]) >= 800, 'min Soldium'
problem1 += lpSum([sodium[f] * foodVars[f] for f in foods]) <= 2000, 'max Sodium'

problem1 += lpSum([carbohydrates[f] * foodVars[f] for f in foods]) >= 130, 'min carbohydrate'
problem1 += lpSum([carbohydrates[f] * foodVars[f] for f in foods]) <= 450, 'max carbohydrate'

problem1 += lpSum([dietaryFiber[f] * foodVars[f] for f in foods]) >= 125, 'min dietaryFiber'
problem1 += lpSum([dietaryFiber[f] * foodVars[f] for f in foods]) <= 250, 'max dietaryFiber'

problem1 += lpSum([Protein[f] * foodVars[f] for f in foods]) >= 60, 'min Protein'
problem1 += lpSum([Protein[f] * foodVars[f] for f in foods]) <= 100, 'max Protein'

problem1 += lpSum([VitA[f] * foodVars[f] for f in foods]) >= 1000, 'min VitA'
problem1 += lpSum([VitA[f] * foodVars[f] for f in foods]) <= 10000, 'max VitA'

problem1 += lpSum([VitC[f] * foodVars[f] for f in foods]) >= 400, 'min VitC'
problem1 += lpSum([VitC[f] * foodVars[f] for f in foods]) <= 5000, 'max VitC'

problem1 += lpSum([Calcium[f] * foodVars[f] for f in foods]) >= 700, 'min Calcium'
problem1 += lpSum([Calcium[f] * foodVars[f] for f in foods]) <= 1500, 'max Calcium'

problem1 += lpSum([Iron[f] * foodVars[f] for f in foods]) >= 10, 'min Iron'
problem1 += lpSum([Iron[f] * foodVars[f] for f in foods]) <= 40, 'max Iron'


```

Then I solve the optimization problem, and show the output by using the syntax as shown below.

```
#Solve the optimization problem
problem1.solve()
LpStatus[problem1.status]

print(problem1)

#Print the output

print("---------The solution to this diet problem is------")
for var in problem1.variables():
    if var.varValue >0:
        if str(var).find('Chosen'):
            print(str(var.varValue)+"units of "+str(var))
print("Total cost of food = $%.2f" % value(problem1.objective))

```

The solution of part 1 optimization problem is shown below.

```
In [2]: problem1.solve()
   ...: LpStatus[problem1.status]
Out[2]: 'Optimal'

In [3]: print("---------The solution to this diet problem is------")
   ...: for var in problem1.variables():
   ...:     if var.varValue >0:
   ...:         if str(var).find('Chosen'):
   ...:             print(str(var.varValue)+"units of "+str(var))
   ...:
   ...: print("Total cost of food = $%.2f" % value(problem1.objective))
---------The solution to this diet problem is------
52.64371units of Foods_Celery,_Raw
0.25960653units of Foods_Frozen_Broccoli
63.988506units of Foods_Lettuce,Iceberg,Raw
2.2929389units of Foods_Oranges
0.14184397units of Foods_Poached_Eggs
13.869322units of Foods_Popcorn,Air_Popped
Total cost of food = $4.34
```

The output showed that the solution is optimal and the total cost is $4.34

Part 2

For part 2 of this question, I added 3 more constraints.

1. The amount of food if chosen must be greater than 10% of portion
2. Celery and Broccoli cannot be chosen at the same time
3. At least three kinds of meat/poultry/fish/eggs must be selected.

Detailed code and comment are shown below:

```
# ================================================================================
# Add constraint if a food is selected, then a minimum of 1/10 serving must be chosen.
# ================================================================================
for f in foods:
    #Add the lower bound to make sure that at least 0.1 of a chosen food
    problem1 += foodVars[f] >= 0.1 * chosenVars[f]
    #Add upper bound to link continous and binary variable
    problem1 += foodVars[f] <= 10000000 * chosenVars[f]
# ================================================================================
# Add constraint so that only one but not both celery and brocolli are selected
# ================================================================================

problem1 += chosenVars['Celery, Raw'] +chosenVars['Frozen Broccoli'] <=1

# ================================================================================
# To get day-to-day variety in protein,
# Add constraint so that at least 3 kinds of meat/poultry/fish/eggs must be selected
# ================================================================================
problem1 += chosenVars['Roasted Chicken'] +chosenVars['Poached Eggs']+ chosenVars['Scrambled Eggs']
+chosenVars['Bologna,Turkey']+ chosenVars['Frankfurter, Beef'] +chosenVars['Ham,Sliced,Extralean']
+chosenVars['Kielbasa,Prk'] +chosenVars['Hotdog, Plain']+ chosenVars['Pork'] +chosenVars['Sardines in Oil']
+ chosenVars['White Tuna in Water'] >=3

```

The solution of part 2 optimization problem is shown below:

```
In [5]: LpStatus[problem1.status]
Out[5]: 'Optimal'

In [6]: problem1.solve()
   ...: LpStatus[problem1.status]
   ...:
   ...: #Print the output
   ...:
   ...: print("---------The solution to this diet problem is------")
   ...: for var in problem1.variables():
   ...:     if var.varValue >0:
   ...:         if str(var).find('Chosen'):
   ...:             print(str(var.varValue)+"units of "+str(var))
   ...:
   ...: print("Total cost of food = $%.2f" % value(problem1.objective))
---------The solution to this diet problem is------
42.399358units of Foods_Celery,_Raw
0.1units of Foods_Kielbasa,Prk
82.802586units of Foods_Lettuce,Iceberg,Raw
3.0771841units of Foods_Oranges
1.9429716units of Foods_Peanut_Butter
0.1units of Foods_Poached_Eggs
13.223294units of Foods_Popcorn,Air_Popped
0.1units of Foods_Scrambled_Eggs
Total cost of food = $4.51
```

The solution(optimal) satisfies all three constraints: only celery is chosen, at least 1/10 serving, and at least 3 kinds of protein which are poached eggs, scrambled eggs and Kielbasa Port. The total cost is $4.51