SOLUTION for 15.2:

For part 1 of the problem, the solution is:

```
Food_List_Lettuce,Iceberg,Raw  63.988506
      Food_List_Celery,_Raw  52.643710
   Food_List_Popcorn,Air_Popped  13.869322
          Food_List_Oranges   2.292939
  Food_List_Frozen_Broccoli   0.259607
     Food_List_Poached_Eggs   0.141844
The total cost of this optimized diet is: $4.34
```

For part 2 of the problem, the solution is:

```
 Food_List_Lettuce,Iceberg,Raw  80.919121
           Food_List_Celery,_Raw  43.154119
    Food_List_Popcorn,Air_Popped  13.181772
               Food_List_Oranges   3.076516
         Food_List_Peanut_Butter   2.046457
       selected_food_Celery,_Raw   1.000000
  selected_food_Popcorn,Air_Popped   1.000000
    selected_food_Poached_Eggs   1.000000
  selected_food_Lettuce,Iceberg,Raw   1.000000
   selected_food_Peanut_Butter   1.000000
          selected_food_Oranges   1.000000
         Food_List_Poached_Eggs   0.141844
The total cost of this optimized diet is: $4.49
```

In [1]:
```
import pulp
from pulp import *
import pandas as pd
import xlrd
```

In [14]:
```
# Read the excel file into a pandas dataframe
diet_df = pd.read_excel(r"diet.xls")[0:64]
```

In [15]:
```
diet_df.head()
```

Out[15]:

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carbohydrates g	Dietary_Fiber g	Protein g
0	Frozen Broccoli	0.16	10 Oz Pkg	73.8	0.0	0.8	68.2	13.6	8.5	8.0
1	Carrots,Raw	0.07	1/2 Cup Shredded	23.7	0.0	0.1	19.2	5.6	1.6	0.6
2	Celery, Raw	0.04	1 Stalk	6.4	0.0	0.1	34.8	1.5	0.7	0.3
3	Frozen Corn	0.18	1/2 Cup	72.2	0.0	0.6	2.5	17.1	2.0	2.5
4	Lettuce,Iceberg,Raw	0.02	1 Leaf	2.6	0.0	0.0	1.8	0.4	0.3	0.2

In [16]:
```
diet_df.shape
```

Out[16]:
```
(64, 14)
```

In [17]:
```
food_list = list(diet_df['Foods'])
food_list[:3]
```

Out[17]:
```
['Frozen Broccoli', 'Carrots,Raw', 'Celery, Raw']
```

In [18]:
```
# Creating dicts of all the decision variables of all food lists

cost_per_serving = dict(zip(food_list,diet_df['Price/ Serving']))
serving_size = dict(zip(food_list,diet_df['Serving Size']))
calories = dict(zip(food_list,diet_df['Calories']))
cholestrol = dict(zip(food_list,diet_df['Cholesterol mg']))
total_fat_g = dict(zip(food_list,diet_df['Total_Fat g']))
sodium_mg = dict(zip(food_list,diet_df['Sodium mg']))
carbs_g = dict(zip(food_list,diet_df['Carbohydrates g']))
fiber_g = dict(zip(food_list,diet_df['Dietary_Fiber g']))
protein_g = dict(zip(food_list,diet_df['Protein g']))
vit_a = dict(zip(food_list,diet_df['Vit_A IU']))
vit_c = dict(zip(food_list,diet_df['Vit_C IU']))
calcium_mg = dict(zip(food_list,diet_df['Calcium mg']))
iron_mg = dict(zip(food_list,diet_df['Iron mg']))
```

In [19]:
```
list_nutrition = [cholestrol,total_fat_g,sodium_mg,carbs_g,fiber_g,protein_g,vit_a,vit_c,calcium_mg,i
```

Now, I am going to greate a Linear Problem object using the LpProblem function in PuLP

In [20]:
```
lin_prob = LpProblem("diet_part_one", LpMinimize)
```

Now, I am going to create a dictionary of food list with boundary for the minimum intake. I am doing this so the intake of a particular food item should be atleast greater than or equal to zero. Without this line of logic, optimization solution might end up resulting in a solution with negative quantity of one or more food item to minimize cost.

In [21]:
```
food_var = LpVariable.dicts(
 "Food_List",
 food_list,
 lowBound=0,
 cat='Continuous'
)
```

THE OBJECTIVE FUNCTION:

In [22]:
```
lin_prob += lpSum(cost_per_serving[e]*food_var[e] for i, e in enumerate(food_list))
```

```
CONSTRAINTS
                    MINIMUM         MAXIMUM
Calories -             1500            2500
Cholesterol mg -         30             240
Total_Fat g -            20              70
Sodium mg -             800            2000
Carbohydrates g -       130             450
Dietary_Fiber g -       125             250
Protein g -              60             100
Vit_A IU -             1000           10000
Vit_C IU -              400            5000
Calcium mg -            700            1500
Iron mg -                10              40
```

In [23]:
```
max_Bound = [2500,240,70,2000,450,250,100,10000,5000,1500,40]
min_Bound = [1500,30,20,800,130,125,60,1000,400,700,10]
list_nutrition = [calories, cholestrol,total_fat_g,sodium_mg,carbs_g,fiber_g,protein_g,vit_a,vit_c,ca
iron_mg]
```

In [24]:
```
# Constraints for Max Boundary (Upper Limits)
for i in range(0,len(max_Bound)):
    lin_prob += lpSum(list_nutrition[i][e] * food_var[e] for j, e in enumerate(food_list)) <= max_Bou
```

In [25]:
```
 # Constraints for Min Boundary (Lower Limits)
for i in range(0,len(min_Bound)):
    lin_prob += lpSum(list_nutrition[i][e] * food_var[e] for j, e in enumerate(food_list)) >= min_Bou
```

In [26]:
```
lin_prob.solve()
```

Out[26]: 1

In [28]:
```
# Printing the status of the solution. In case the problem is ill-formulated or there is not sufficie
# the solution may be infeasible or unbounded.
print("Status:", LpStatus[lin_prob.status])
```

```
Status: Optimal
```

In [29]:
```
food_name = []
food_val = []

for v in lin_prob.variables():
    if v.varValue>=0:
        food_name.append(v.name)
        food_val.append(v.varValue)
```

In [30]:
```
df1 = pd.DataFrame(list(zip(food_name, food_val)), columns =['Name', 'val'])
df1 = df1.sort_values(by='val', ascending=False)
```

In [31]:
```
df1.head(n=10)
```

Out[31]:

	Name	val
28	Food_List_Lettuce,Iceberg,Raw	63.988506
11	Food_List_Celery,_Raw	52.643710
40	Food_List_Popcorn,Air_Popped	13.869322
35	Food_List_Oranges	2.292939
20	Food_List_Frozen_Broccoli	0.259607
39	Food_List_Poached_Eggs	0.141844
43	Food_List_Potatoes,_Baked	0.000000
46	Food_List_Rice_Krispies	0.000000
45	Food_List_Raisin_Brn,_Kellg'S	0.000000
44	Food_List_Pretzels	0.000000

Table of a list of foods with quantity > 0.

In [57]:
```
print(df1[df1['val'] > 0])
```

```
                            Name        val
28  Food_List_Lettuce,Iceberg,Raw  63.988506
11          Food_List_Celery,_Raw  52.643710
40   Food_List_Popcorn,Air_Popped  13.869322
35              Food_List_Oranges   2.292939
20      Food_List_Frozen_Broccoli   0.259607
39         Food_List_Poached_Eggs   0.141844
```

So, the optimal solution is to eat 63.988506 servings of raw iceberg Lettuce, 52.643710 servings of raw celery, 13.869322 of air popped pop corn, 2.292939 servings of oranges, 0.259607 servings of frozen broccoli, and 0.141844 servings of poached eggs.

Next, let us look at the cost

In [34]:
```
print("The total cost of this optimized diet is: ${}".format(round(value(lin_prob.objective),2)))
```

```
The total cost of this optimized diet is: $4.34
```

Now, the second part of the problem:

In [80]:
```
diet_df = pd.read_excel(r"diet.xls")[0:64]

food_list = list(diet_df['Foods'])

cost_per_serving = dict(zip(food_list,diet_df['Price/ Serving']))
serving_size = dict(zip(food_list,diet_df['Serving Size']))
calories = dict(zip(food_list,diet_df['Calories']))
cholestrol = dict(zip(food_list,diet_df['Cholesterol mg']))
total_fat_g = dict(zip(food_list,diet_df['Total_Fat g']))
sodium_mg = dict(zip(food_list,diet_df['Sodium mg']))
carbs_g = dict(zip(food_list,diet_df['Carbohydrates g']))
fiber_g = dict(zip(food_list,diet_df['Dietary_Fiber g']))
protein_g = dict(zip(food_list,diet_df['Protein g']))
vit_a = dict(zip(food_list,diet_df['Vit_A IU']))
vit_c = dict(zip(food_list,diet_df['Vit_C IU']))
calcium_mg = dict(zip(food_list,diet_df['Calcium mg']))
iron_mg = dict(zip(food_list,diet_df['Iron mg']))
list_nutrition = [cholestrol,total_fat_g,sodium_mg,carbs_g,fiber_g,protein_g,vit_a,vit_c,calcium_mg,i
```

In [81]:
```
lin_prob2 = LpProblem("diet_part_two", LpMinimize)
```

In [82]:
```
selected_food = LpVariable.dicts("selected_food",food_list,0,1,cat='Integer')
```

In [83]:
```
food_var = LpVariable.dicts(
    "Food_List",
    food_list,
    lowBound=0,
    cat='Continuous'
)
```

In [84]:
```
# Objective Function:

lin_prob2 += lpSum(cost_per_serving[e]*food_var[e] for i, e in enumerate(food_list))
```

In [85]:
```
# Lower Bounds:

lin_prob2 += lpSum(calories[e] * food_var[e] for i, e in enumerate(food_list)) >= 1500
lin_prob2 += lpSum(cholestrol[e] * food_var[e] for i, e in enumerate(food_list)) >= 30
lin_prob2 += lpSum(total_fat_g[e] * food_var[e] for i, e in enumerate(food_list)) >= 20
lin_prob2 += lpSum(sodium_mg[e] * food_var[e] for i, e in enumerate(food_list)) >= 800
lin_prob2 += lpSum(carbs_g[e] * food_var[e] for i, e in enumerate(food_list)) >= 130
lin_prob2 += lpSum(fiber_g[e] * food_var[e] for i, e in enumerate(food_list)) >= 125
lin_prob2 += lpSum(protein_g[e] * food_var[e] for i, e in enumerate(food_list)) >= 60
lin_prob2 += lpSum(vit_a[e] * food_var[e] for i, e in enumerate(food_list)) >= 1000
lin_prob2 += lpSum(vit_c[e] * food_var[e] for i, e in enumerate(food_list)) >= 400
lin_prob2 += lpSum(calcium_mg[e] * food_var[e] for i, e in enumerate(food_list)) >= 700
lin_prob2 += lpSum(iron_mg[e] * food_var[e] for i, e in enumerate(food_list)) >= 10

lin_prob2 += lpSum(calories[e] * food_var[e] for i, e in enumerate(food_list)) <= 2500
lin_prob2 += lpSum(cholestrol[e] * food_var[e] for i, e in enumerate(food_list)) <= 240
lin_prob2 += lpSum(total_fat_g[e] * food_var[e] for i, e in enumerate(food_list)) <= 70
lin_prob2 += lpSum(sodium_mg[e] * food_var[e] for i, e in enumerate(food_list)) <= 2000
lin_prob2 += lpSum(carbs_g[e] * food_var[e] for i, e in enumerate(food_list)) <= 450
lin_prob2 += lpSum(fiber_g[e] * food_var[e] for i, e in enumerate(food_list)) <= 250
lin_prob2 += lpSum(protein_g[e] * food_var[e] for i, e in enumerate(food_list)) <= 100
lin_prob2 += lpSum(vit_a[e] * food_var[e] for i, e in enumerate(food_list)) <= 10000
lin_prob2 += lpSum(vit_c[e] * food_var[e] for i, e in enumerate(food_list)) <= 5000
lin_prob2 += lpSum(calcium_mg[e] * food_var[e] for i, e in enumerate(food_list)) <= 1500
lin_prob2 += lpSum(iron_mg[e] * food_var[e] for i, e in enumerate(food_list)) <= 40
```

Constraint A

If a food is selected, then a minimum of 1/10 serving must be chosen.

Also, Giving importance to food_var only if the selected_food variable is 1

Setting an upper bound of 10000 and lower bound of 0.1,

In [86]:
```
ub = 10000
lb = 0.1

for i in food_list:
    lin_prob2 += food_var[i] >= lb * selected_food[i]
    lin_prob2 += food_var[i] <= ub * selected_food[i]
```

Constraint B

Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected. This can be done by making sure that the sum of binary variables (lettuce and broccoli) = 1. This ensures that only one can be selected.

In [87]:
```
lin_prob2 += selected_food['Frozen Broccoli']+selected_food['Celery, Raw'] <=1
```

Constraint C

To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected.

Well, I tried to do this using regex. Couldnt figure it out. So, TA or Peers, whoever is grading this code, if you can leave a comment about using regex here, that'd help me learn. Thanks.

In [78]:
```
lin_prob2 += selected_food['Roasted Chicken'] + selected_food['Chicknoodl Soup'] + selected_food['Fra
+ selected_food['Vegetbeef Soup'] + selected_food['Poached Eggs'] + selected_food['Scrambled Eggs']
+ selected_food['Bologna,Turkey'] + selected_food['Ham,Sliced,Extralean'] + selected_food['Hamburger
+ selected_food['Splt Pea&Hamsoup'] + selected_food['Hotdog, Plain'] + selected_food['Pizza W/Peppero
+ selected_food['Hamburger W/Toppings'] + selected_food['Pork'] + selected_food['Sardines in Oil']
+ selected_food['White Tuna in Water'] + selected_food['Neweng Clamchwd'] + selected_food['Tomato Sou
+ selected_food['New E Clamchwd,W/Mlk'] >= 3
```

Out[78]:
```
1*selected_food_New_E_Clamchwd,W_Mlk + -3 >= 0
```

In [88]:
```
lin_prob2.solve()
```

Out[88]:
```
1
```

In [89]:
```
# Printing the status of the solution. In case the problem is ill-formulated or there is not sufficie
# the solution may be infeasible or unbounded.

print("Status:", LpStatus[lin_prob2.status])
```

```
Status: Optimal
```

In [90]:
```
food_name2 = []
food_val2 = []
for v in lin_prob2.variables():
    if v.varValue>=0:
        food_name2.append(v.name)
        food_val2.append(v.varValue)
```

In [91]:
```
df2 = pd.DataFrame(list(zip(food_name2, food_val2)), columns =['Name', 'val'])
df2 = df2.sort_values(by='val', ascending=False)
```

In [92]:
```
print(df2[df2['val'] > 0])
```

```
                                  Name        val
28       Food_List_Lettuce,Iceberg,Raw  80.919121
11               Food_List_Celery,_Raw  43.154119
40        Food_List_Popcorn,Air_Popped  13.181772
35                   Food_List_Oranges   3.076516
36              Food_List_Peanut_Butter   2.046457
75           selected_food_Celery,_Raw   1.000000
104   selected_food_Popcorn,Air_Popped   1.000000
103         selected_food_Poached_Eggs   1.000000
92   selected_food_Lettuce,Iceberg,Raw   1.000000
100        selected_food_Peanut_Butter   1.000000
99               selected_food_Oranges   1.000000
39              Food_List_Poached_Eggs   0.141844
```

Cost of this optimized diet is:

In [93]:
```
print("The total cost of this optimized diet is: ${}".format(round(value(lin_prob2.objective),2)))
```

```
The total cost of this optimized diet is: $4.49
```