Name: XXX

Problem Set 10

Learning Objective:

- Create Python code to automate a given task.
- Formulate linear optimization models to inform a business decision.

Overview:

This problem set assesses your ability to turn an abstract formulation into reusable optimization software, as discussed in Week 12.

Grading

There are three possible scores you can get from submitting this assignment on time (submitting a blank file or one without any apparent effort does not count). Note that the rubric is designed to incentivize you to go for 100% mastery of the material, as the little details matter a lot in business analytics.

Grade	Description
5 out of 5	Perfect submission with no significant errors.
4 out of 5	Near perfect submission with one or more significant errors.
2 out of 5	Apparent effort but far from perfect.

Q1. Reusable Software for Assortment Planning

Create a function called "optimizeAssortment" with two input arguments:

- inputFile: filename of the input file. See the attached "PS10-books-input-1.xlsx" and "PS10-books-input-2.xlsx" for examples of the sample input.
- outputFile: filename of the output file that the function will create.

The function should implement the abstract formulation for Problem 9.2 and produce the outputFile. See the attached "PS10-books-sampleOutput-1.xlsx" for the desired output file format corresponding to the "PS10-books-input-1.xlsx" input file. The abstract formulation is reproduced below for your convenience.

Data:

- B: the set of books.
- G: the set of genres.
- a_{bg}: whether book b is in genre g.
- q_g: how many books we need of genre g.

Decision Variables: Let x_b deibite whether to carry book b. (Binary)

Objective and constraints:

$$\begin{aligned} \text{Minimize:} \quad & \sum_{b \in B} x_b \\ \text{subject to:} \quad & \\ \text{(Enough books in genre)} \quad & \sum_{b \in B} a_{bg} x_b \geq q_g \quad \text{for each genre } g \in G. \end{aligned}$$

Q2. Assigning of Final Grades

This question asks you to create software that can help a professor assign final grades in such a way so that the average GPA rounds to 3.5, while obtaining an assignment in which there are gaps in scores between consecutive grade levels, and no particular grade is assigned to disproportionally many students.

Data:

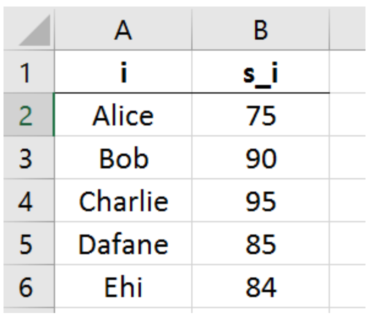
- I: the set of students.
- n: the number of students.
- $J = \{0, 1, \cdots\}$: numerical indices used to denote the various grade levels.
- s_i: the overall score of student $i \in I$ (between 0 and 100).
- g_j: the GPA corresponding to grade level $j \in J$.

Decision Variables:

- x_{ij}: whether to assign student i to grade level j. (Binary)
- t_j: the number of students assigned to grade level j. (Continuous)
- L_j: the score cutoff for grade level j. (Continuous)
- U_j: the maximum score in grade level j. (Continuous)

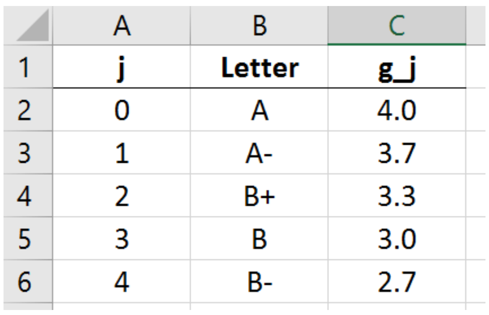
$$
\begin{array}{rrcl}
\text{Min} & \displaystyle\sum_{j \in J} (U_j - L_j) + 0.1 \sum_{j \in J} t_j \times t_j & & \\
\text{s.t.} & & & \\
\text{(Average GPA)} & \displaystyle 3.495n \le \sum_{i \in I} \sum_{j \in J} x_{ij} g_j \le 3.505n & & \\
\text{(Assignment)} & \displaystyle \sum_{j \in J} x_{ij} = 1 & & \text{for each } i \in I. \\
\text{(Max score)} & s_i x_{ij} \le U_j & & \text{for each } i \in I, j \in J. \\
\text{(Min score)} & 100(1 - x_{ij}) + s_i x_{ij} \ge L_j & & \text{for each } i \in I, j \in J. \\
\text{(Correct totals)} & \displaystyle \sum_{i \in I} x_{ij} = t_j & & \text{for each } j \in J. \\
\text{(Bounds)} & L_j \le U_j & & \text{for each } j \in J. \\
\text{(Ordering)} & U_j \le L_{j-1} & & \text{for each } j \in J \text{ with } j \ge 1.
\end{array}
$$

The input data is contained in an Excel file named `PS10-grade-input.xlsx` with two sheets. The first sheet, named "Scores", contains the score of each student. The first five entries look like:



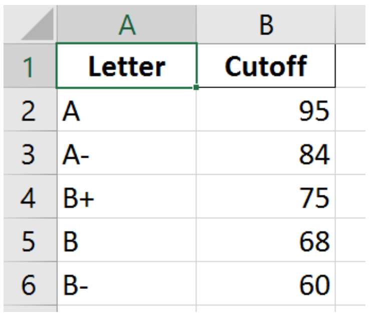
	A	B
1	**i**	**s_i**
2	Alice	75
3	Bob	90
4	Charlie	95
5	Dafane	85
6	Ehi	84

The second sheet, named "Levels", is as follows



	A	B	C
1	j	Letter	g_j
2	0	A	4.0
3	1	A-	3.7
4	2	B+	3.3
5	3	B	3.0
6	4	B-	2.7

The output data should be an Excel file named `PS10-grade-output.xlsx` that contains the cutoff (L_j) for each grade level j. It should look like



	A	B
1	Letter	Cutoff
2	A	95
3	A-	84
4	B+	75
5	B	68
6	B-	60

Q3. Team Assignment

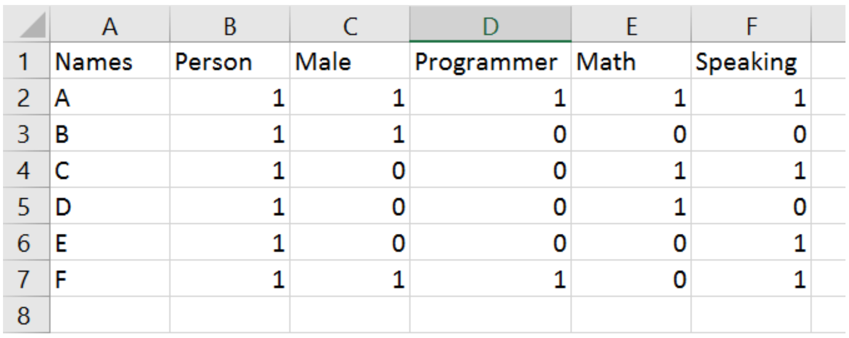
The following MIP can used to assign students into project teams to balance the overall characteristics of each team.

Data:

- I: set of students.
- n: number of teams
- $J = \{1, 2, \cdots, n\}$: set of teams.
- K: set of characteristics.
- a_{ik}: student i's value for characteristics k.
- w_k: the weight for characteristics k in the objective.
- L_k: the ideal lower bound for the sum of characteristic k for any team.
- U_k: the ideal upper bound for the sum of characteristics k for any team.

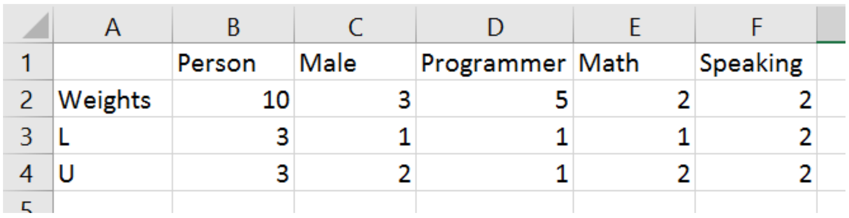
You should assume that the data is given in a excel file with the same format as the `PS10-Team-input-1.xlsx` and `PS10-Team-input-2.xlsx` files attached to this assignment.

The sheet named "Students" encodes I, K and a_{ik}'s. In the below screenshot of `PS10-Team-input-1.xlsx`, $I = \{A, B, C, D, E, F\}$, and $K = \{Person, Male, Programmer, Math, Speaking\}$.



	A	B	C	D	E	F
1	Names	Person	Male	Programmer	Math	Speaking
2	A	1	1	1	1	1
3	B	1	1	0	0	0
4	C	1	0	0	1	1
5	D	1	0	0	1	0
6	E	1	0	0	0	1
7	F	1	1	1	0	1
8						

The sheet named "Parameters" encodes the w_k, L_k and U_k for each characteristic k.



	A	B	C	D	E	F
1		Person	Male	Programmer	Math	Speaking
2	Weights	10	3	5	2	2
3	L	3	1	1	1	2
4	U	3	2	1	2	2
5						

Decision variables:

- x_{ij} : whether to assign student i to team j. (Binary)
- s_k : maximum deviation below the ideal lower bound L_k for characteristic k. (Continuous)
- t_k : maximum deviation above the ideal upper bound U_k for characteristic k. (Continuous)

Objective and constraints:

$$
\begin{array}{rrl}
 & \text{Minimize:} & \sum_{k \in K} w_k(s_k + t_k) \\
 & \text{subject to:} & \\
\text{(Every person assigned)} & \sum_{j \in J} x_{ij} = 1 & \text{For each person } i \in I. \\
\text{(Team balance)} & L_k - s_k \leq \sum_{i \in I} a_{ik} x_{ij} \leq U_k + t_k & \text{For each team } j \in J \text{ and each } k \in K. \\
\text{(Non-negativity)} & s_k, t_k \geq 0 & \text{for all } k.
\end{array}
$$

Write a function called "assignTeams" with the following input arguments:

- **inputFile:** path to the input spreadsheet.
- **n:** the number of teams to divide students into.

The function should return two variables:

- **df:** a DataFrame with one column called "Team". The index should be the name of each individual, and the column "Team" should specify the number j to which the person is assigned.
- **objval:** the optimal objective value.