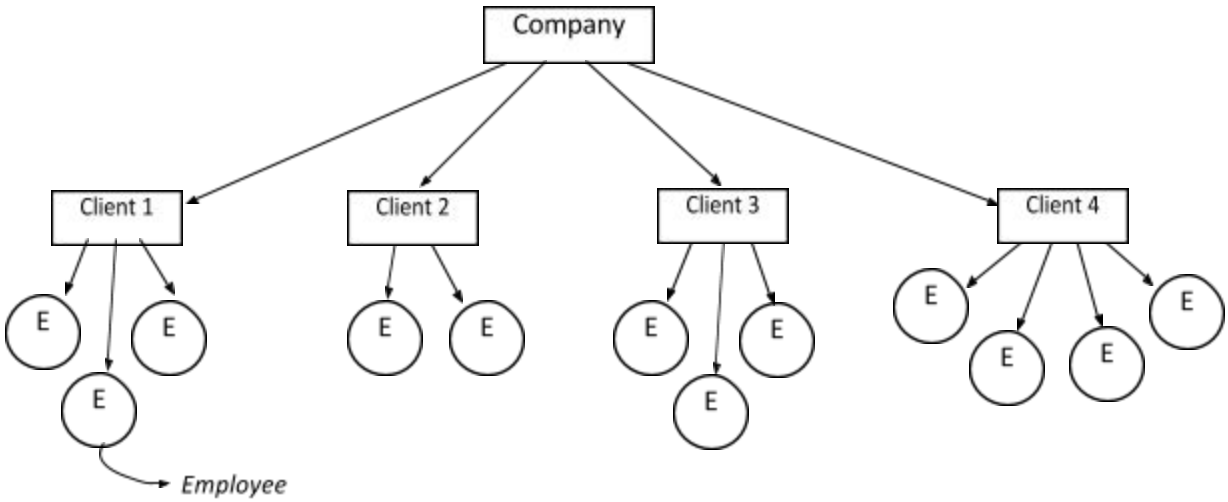
A solução deverá ser implementada utilizando C#, Java ou C++. Procure utilizar as melhores práticas de desenvolvimento, padrões de projeto e testabilidade (boa performance é desejável). Evite utilizar bibliotecas externas, exceto para testes unitários.

O código fonte deve ser enviado em formato .zip (remover qualquer arquivo .exe), juntamente com as instruções de execução.

Company Employees Balancing

A call center company is needing a solution to improve its policies of career progression and team allocation. This company has some clients and each client has a team, like illustrated below:



A team can have any amount of employees. Each employee has a professional level (PLevel) ranging from 1 to 5. Level 1 is the beginner and the level 5 is the most senior. In addition, the employee also has this information:

- Birth year
- Admission year
- Last progression year (If the employee never receive a progression, this date is equal the admission year)

The new application need to calculate the next promoted employees, in accordance with the following criteria and restrictions. **The employee needs to meet all the restrictions to be eligible for promotion**.

Criteria	**Restriction**
Company time	Minimum 1 year
Time without progression	Minimum 2 years, if level is 4.
Age	No Restrictions.

Criteria	**Criteria Weight**
Company time	2 points per year
Time without progression	3 points per year
Age	1 point per 5 years old

The employees must be ranked by the sum of points calculated by these criterias, if restrictions are met. The user will inform the number of employees he wants to promote. Initially, the application is in the current year. After any progression applied, the application adds one year to current year.

The company has a rule to apply in team composition. Each team needs to be composed by enough employees to meet at least its minimum maturity points (sum of PLevels). For example, if a team needs a maturity of 12, the sum of PLevels from its employees must be of 12 or more:

- 2 employees level 4
- 1 employee level 3
- 1 employee level 2

Or

- 1 employee level 5
- 1 employee level 4
- 1 employee level 3

After a progression, some teams may have its actual maturity a lot higher than needed comparing with others teams. For example Team A needs a maturity of 5 but it has 10, Team B needs 7 and it has 8. It would be more balanced if they had 8 (Team A) and 10 (Team B), 3 extra points for each team. The application needs a feature to balance the employees between teams to obtain the best team organization.

The application details are specified below:

Features

1. **Load input data (load)**
   Loads the data to be used by the application in CSV formatted files.

2. **Allocate employees to teams (allocate)**
   Allocates all employees between the teams given on the input accordingly to their maturity needs.

3. **Promote employees (promote)**
   The user informs the number of employees that should be promoted. The application should evaluate the restrictions and criteria of employees and return the list of promoted ones. This list should be applied and shown to the user.

4. **Balance Teams (balance)**
   Move employees between teams to achieve the best balance. Consider as an extra maturity points of a team the amount that exceeds the minimum maturity specified. The best scenario is when the subtraction of the highest extra maturity points and the lowest is as close to 0 (zero) as possible.

   Example:
   Team A - Min Maturity 7 - Current Maturity 10 - Extra Maturity Points 3 (10 - 7)
   Team B - Min Maturity 10 - Current Maturity 12 - Extra Maturity Points 2 (12 - 10)
   Team C - Min Maturity 13 - Current Maturity 15 - Extra Maturity Points 2 (15 - 13)
   Result -> Team A (3) - Team B (2) = 1

The solution must be a console application that should work as described below.

Inputs

The application should receive the input as two separate txt files on the load command.

File 1 - Teams

Team Name,Team Min. Maturity
Client 1,3
Client 2,5
Client 3,7

File 2 - Employees

Employee Name,PLevel,Birth Year,Admission Year,Last Progression Year
Diego,3,1991,2011,2015
Lucas,4,1990,2012,2017
Luciano,1,1979,2011,2013
Luiz,4,1980,2005,2014
Daniel,2,1989,2013,2015
Jose Carlos,2,1994,2013,2016
Felipe,3,1976,2018,2018

Console Outputs

The combination of employees on the teams is not relevant. It's only required that the allocation respects the minimum maturity per team and the balance command distributes employees to achieve the best maturity for each team.

Commands are in **BOLD.**

load "team.txt" "employees.txt"

loaded

allocate

Client 1 - Min. Maturity 3 - Current Maturity 7
Felipe - 3
Daniel - 2
José Carlos - 2

Client 2 - Min. Maturity 5- Current Maturity 5
Luiz - 4
Luciano - 1

Client 3 - Min. Maturity 7 - Current Maturity 7
Lucas - 4
Diego - 3

promote 2

Luciano - From: 1 - To: 2
Luiz - From: 4 - To: 5

balance

Client 1 - Min. Maturity 3 - Current Maturity 5
Felipe - 3
Daniel - 2

Client 2 - Min. Maturity 5- Current Maturity 7
José Carlos - 2
Diego - 3
Luciano - 2

Client 3 - Min. Maturity 7 - Current Maturity 9
Lucas - 4
Luiz - 5

promote 4

Luciano - From: 2 - To: 3
Daniel - From: 2 - To: 3
Jose Carlos - From: 2 - To: 3
Diego - From: 3 - To: 4

balance

Client 1 - Min. Maturity 3 - Current Maturity 6
Felipe - 3
Daniel - 3

Client 2 - Min. Maturity 5- Current Maturity 8
Lucas - 4
Diego - 4

Client 3 - Min. Maturity 7 - Current Maturity 11
Luiz - 5
Luciano - 3
Jose Carlos - 3