

Laboratory Exercise

Script programs are useful for text processing involving many files. Write a **bash** shell script called **minileague** to input the scores of soccer games and draw up the league table among a few selected teams. The script will first read in the team information from a file whose name is specified as the first parameter, e.g. **big6.dat**. Each line of the file contains the short name of a team, followed by the name of the team. Your program then reads in a sequence of files containing the scores of games on different days or weeks, as indicated by the **prefix** of those files specified as the second parameter, e.g. **score2122**. This concept of a mini-league is useful for some common tie-breaking rules.

In a soccer game, there could be one winning team and one losing team, or both teams end up with a draw. The winning team will score 3 points, the losing team will score 0 point. If the game ends up with a draw, both teams will score 1 point. The ranking criterion is according to the number of points earned by a team. The champion is the one scoring the highest number of points. In case of a tie among two or more teams, the number of points earned by the teams in a Mini-League (i.e. just consider the games among those teams) will be used to rank them. Note that your program **minileague** could be used to perform this ranking criterion on the subset of teams involved in the tie-breaking situation. If there is still a tie among the teams, tie breaking will be based on the goal difference and the team with a larger goal difference will rank higher. In case of a tie in goal difference, tie breaking will be based on the number of goals scored and the team with a higher number of scored goals will rank higher. If there is still a tie after using the goal difference and goal scored, both teams will be considered to rank equally.

Note that for a double round-robin league common in most countries, each team needs to meet each other team twice, once at home and once away. If the number of teams, n , is even, each round will involve all n teams in $n/2$ games, and we need $n-1$ rounds for the first round-robin, and then another $n-1$ rounds for the second round-robin. If n is odd, there will be one team enjoying a *bye* round, and only $(n-1)/2$ teams are involved. A total of n rounds are needed for the first round-robin, and the same for the second round-robin.

You can assume that there is no error in the information contained in each file. In other words, you do not need to handle errors in the configuration file and score files.

For instance, the program can be run like:

```
minileague big6.dat score2122
```

Here, **score2122** is the prefix of the files containing the game scores, e.g. **score21220815**, **score21220821**. Each line of the score file contains the score of a game, the first team is the home team and the second team is the visiting team. You may consider using the back quote operator in a for-loop to extract all information. Typical file contents are shown below:

Filename	Content
big6.dat	MCI ManchesterCity LIV Liverpool CHE Chelsea MUN ManchesterUnited ARS Arsenal TOT Tottenham
score21220815	TOT 1 MCI 0 NEW 2 WHU 4 NOR 0 LIV 3
score21220821	LIV 2 BUR 0 AVL 2 NEW 0 CRY 0 BRE 0 LEE 2 EVE 2 MCI 6 NOR 0

The output of your program should look like this, after processing upon the recent files till late Jan in this season (*formatting is not important*):

Mini-league with 6 teams										
Rank	Team	G	W	D	L	Point	GF	GA	GD	
1	ManchesterCity	7	5	1	1	16	13	4	9	
2	Chelsea	8	3	3	2	12	11	6	5	
3	Liverpool	6	2	4	0	10	16	7	9	
4	ManchesterUnited	5	2	1	2	7	7	10	-3	
5	Tottenham	6	1	1	4	4	4	13	-9	
6	Arsenal	6	1	0	5	3	6	17	-11	

Note that if all 20 teams are included in the configuration file, it would generate the full league table. Here, the mini-league table is sorted according to the points, followed by tie-breaking rules specified above. In case that some teams are equal in ranking, order them alphabetically according to their names.

Remember that the program is supposed to work on *any correct datasets*. As a result, you ***should not*** hardcode team names in the program. Furthermore, it is ***not required*** that you use the real data set for program development and testing. It would be easier if you use a smaller number of teams with fewer results for development and initial testing. Here is an example based on simple testing cases. You could also *change* the scores for the teams to *test* for different computation and ranking scenarios.

Filename	Content
top4.dat	ARS Arsenal CHE Chelsea LIV Liverpool MUN ManchesterUnited
score21220101	LIV 1 CHE 1 MUN 3 ARS 2
score21220102	MUN 0 LIV 5 ARS 0 CHE 2

Mini-league with 4 teams										
Rank	Team	G	W	D	L	Point	GF	GA	GD	
1	Liverpool	2	1	1	0	4	6	1	5	
2	Chelsea	2	1	1	0	4	3	1	2	
3	ManchesterUnited	2	1	0	1	3	3	3	7	-4
4	Arsenal	2	0	0	2	0	2	5	-3	

Requirements:

1. Elementary level: your script can read in the game scores relevant to the teams involved and echo the relevant information correctly.
2. Basic level: your script can produce the correct output with information correctly computed, in the absence of any tie-breaking implementation.
3. Required level: your script can produce the correct sorted output with information correctly computed, with tie-breaking rules working correctly for most scenarios.
4. Bonus level: your script can produce ***all*** correct outputs as well as the detection of at least one form of anomaly, e.g. team A is playing team B three times (instead of two times), team A is hosting team B two times (instead of one at home and one away).

Please provide ***appropriate comments*** and check to *ensure* that your program can be ***executed*** properly under the Linux (**apollo** or **apollo2**) environment before submission.

Name your program **minileague** (note that you cannot submit a file with extension **.sh** in BlackBoard for security reason) and **submit it via BlackBoard on or before 18 February 2022**.