CSCI 2073 – Programming Assignment 4 – Fall 2018

You have been hired to develop a Java program to keep track of team statistics for a soccer league. As game scores are reported, the program is to keep track of team records and related statistics. Your assignment is to develop the **LeagueStats** class, as described below. Game scores will be stored in a comma-separated (.csv) file. Each line contains the following information about a game, with items being separated by commas:

```
Home Team,Away Team,Home Goals,Away Goals
```

You are free to store team and game information in any suitable data structure(s) of your choice, but keep in mind that the actual file may be much larger than the one provided for testing.

The **LeagueStats** class is required to contain the following public methods:

`LeagueStats(String csvFile)`: constructor to read csv file and store game information

`String getStats(String statsFile)`: method to process a file containing a number of requests for league statistics. The file contains one request per line. The possible requests are:

```
STATS team
BEST
HSCORING
RANK team
```

Each request requires that the program produce specific output on a line by itself, as follows:

- STATS: display the team name, wins, draws, losses, goals for, goals against, and points (3 pts. per win, 1 pt. per draw) using the format below (if the team does not exist, the statistics are replaced by the phrase NOT FOUND).
  ```
  TEAM: Name                W: xx D: yy L: zz GF: ww GA: vv PTS: pp
  ```
- BEST: display the same information as shown in the STATS transaction for the team with the largest number of points. If two or more teams are tied with the largest number of points, the tiebreakers are:
  - largest goal differential (the difference between goal scored and goals allowed)
  - most goals scored
  ```
  BEST: Name                W: xx D: yy L: zz GF: ww GA: vv PTS: pp
  ```
- HSCORING: display the names of any teams that have scored more goals than the BEST team using the format below (or NONE if no such team exists).
  ```
  HIGH SCORING: teamX, teamY, …, teamZ
  ```
- RANK: display the team name and its ranking, using the format below (if the team does not exist, the statistics are replaced by the phrase NOT FOUND). Use the same criteria for tie-breakers as used for the BEST operation.
  ```
  RANK: Name x out of y
  ```

Sample input file contents and corresponding output:

`STATS Borussia Dortmund`	`TEAM: Borussia Dortmund  W: 0  D: 1  L: 2  GF: 6  GA: 12  PTS: 1`
`STATS Schalke 04`	`TEAM: Schalke 04         W: 3  D: 0  L: 0  GF: 11  GA: 3  PTS: 9`
`STATS ULM Warhawks`	`TEAM: ULM Warhawks NOT FOUND`
`BEST`	`BEST: Schalke 04         W: 3  D: 0  L: 0  GF: 11  GA: 3  PTS: 9`
`HSCORING`	`HIGH SCORERS: 1. FC Koeln`
`RANK Borussia Dortmund`	`RANK: Borussia Dortmund is ranked 15 out of 16`

The *LeagueTest.java* program, as well as the files *games1-24.csv* and *trans.txt* are provided for basic testing. The sample output above should result from executing *LeagueTest* with your solution and the data files provided. Once you are satisfied with your results, submit LeagueStats.java and any other .java files developed as part of your solution. Do not submit any of the files provided by the instructor for testing.