# Tutorial 5: Lists and Sports Teams

CS 135 Fall 2007

October 10-12, 2007

This week's tutorial is the first to make use of *unbounded data* in Scheme (using lists). We will also be using some starter code (t5-starter.scm) for the first time. This should allow us to get a little more done by having some function and data definitions given beforehand. The material here comes from lecture module 5 of the course notes.

Before beginning, examine the starter code given. The struct `team` holds the name of a team and the number of wins and losses that team has (assume the rules of the sport don't allow ties to occur). Read the comments for the function `win-pct` so that you know what it does and how it works.

## 1 Lists of teams

### 1.1 Total Games Played

Write a function `games-played` which consumes a (possibly empty) list of teams and produces the total number of games played by all teams. This sum should *include* all repeats (that is, ignore the fact that one team's win is likely another team's loss).

### 1.2 Make a list of teams

Given a list of team names (strings), write a function `names->teams` that produces a list of teams corresponding to the names, all with blank records (i.e. no wins and no losses).

### 1.3 Add a game

Since we are only counting the number of wins and losses, we can completely describe the outcome of a game just by giving the name of the team that won and the name of the one that lost.

So write a function `add-game` that consumes two strings for the name of the winning and losing teams (respectively), as well as a list of teams, and produces a new list of teams with the records of the winning and losing teams updated appropriately. You may assume that there is a team in the given list with each of the given names.

### 1.4 And the winners are...

We say a team has a *winning record* if its win percentage (as computed by the `win-pct` function) is strictly greater than 1/2. Note that at least one, but not all, of the teams in a single league can have a winning record.

Write a function `winning-teams` that consumes a list of teams and produces a list of the *names* of the teams with winning records.

### 1.5 Rivals

When two teams become entrenched in a bitter and long-standing rivalry, sometimes their fans care most about which team is doing better rather than their team's overall success.

In order to assist such fans, write a function `rival-winner` that consumes the names of two rival teams, and a list of teams (which includes the two named ones) and produces the name of the team out of the two with the higher win percentage.

## 2 Who have you been playing with?

The following problem is a bit more complex than the previous ones, and will probably require at least one or two helper functions. So use the design recipe, think and plan carefully!

Given a list of teams, we want to determine whether the teams could form a closed league — that is, they have only played each other. Note that this is the case if and only if the following two conditions are met:

1. The total number of games won equals the total number of games lost.

2. The total number of games played by any single team does not exceed the total number of games played by all the other teams combined.

So write a function `same-league?` which consumes a list of teams and produces true iff the given records could have been accumulated just from the teams playing each other.

# 3 Survey

Now that we've had a little over a month of CS 135, including assignments, tutorials, lectures, and an exam, you've probably formed some opinions about various aspects of the course. And it's still early enough that we might be able to change some things to help you learn better.

So we'd like you all to take this little survey. You don't need give your name or any identifying information, and the survey is completely optional as well. In fact, this has NOT been cleared by the office of ethics, so if that makes you uncomfortable, please don't feel pressured to participate.

But thanks very much for those who do choose to participate! We will read all of your responses, and if there is any particular issue that receives widespread attention, we may make some changes to address the concern.

1. What has been the most difficult part of the course to understand? Do you think the material is just really tough or that it could have been taught more clearly?

2. What have you enjoyed most about the course so far (besides the stunningly good looks of your instructors)?

3. What aspect of the tutorials has been most helpful to you in preparing for assignments and exams? What has been least helpful?

4. How is the pace of the tutorials (too fast/too slow/just right)?

5. Do you ever do the extra problems in tutorial and/or look at the solutions? Why or why not?

6. Any other comments or random musings?