SEng 265 — Introduction to Software Engineering
Fall 2002
Assignment No. 4

Note 1 This assignment is to be done, optionally, in teams of 2.

Note 2 Different implementations of the C compiler tend to vary slightly. Your assignments will be tested on Software Engineering's main server (SENG's aserver.seng.engr.uvic.ca), and it is recommended that you do your assignments either on the SENG workstations (located in ELW A321) or using putty (see the resources web page) to connect to aserver. If you use some other Unix system, you may get different results; this will result in mark deductions.

- Due date: Wednesday, Nov. 20, 2001, at the beginning of the class.
- This assignment is worth 20 points.
- This assignment is worth 5% of your total course mark.
- Clearly mark student numbers on all submissions.

Objectives

After completing this assignment, you will have:

- gained experience reading and maintaining code,
- learned dynamic memory allocation in C,
- worked with multiple source files.

Introduction

The program we wrote for assignment 3 has three main restrictions:

1. It was able to handle at most 256 teams.
2. The name of a team is restricted to be at most 80 characters long.
3. It could only deal with one tournament

Your Task

We are going to lift all three restrictions. Your task is to modify a solution to assignment 3 that I wrote. You should implement the following features:

1. Dynamically allocate the space for each team (instead of having to use arrays of 80 characters long for every team name, regardless of its actual length). For each new team in the tournament, you should allocate *exactly* the required bytes.
2. Dynamically allocate the space needed for the teams. Instead of an array of structs, you are going to use a linked list of structs. You should be able to read as many different teams as necessary as long as there is enough memory available to handle them.

3. You should support several inputs per run. Several tournaments could be included in an input file, one after another, with no separator in between them. You should restart the competition with the new data. For instance, the input file will look like:

```
2
Manchester United
Real Madrid
2,3
2
Barcelona
Manchester United
0,1
```

4. At the end of the tournament, you should report statistics about all the tournaments. For every team that plays in the tournament, you should report: Name, Percentage, Wins, Loses, Total Scored, Total Against. The teams should be reported from best to worse (according to their performance). For instance, for the previous input, the output should be:

```
Tournament 1:
Round 1:
Real Madrid defeats Manchester United by score 3 2
Champion: Real Madrid
2
Manchester United
Real Madrid
2,3
2
Barcelona
Manchester United
0,1
Tournament 2:
Round 1:
Manchester United defeats Barcelona by score 1 0
Champion: Manchester United
Statistics:
Team              | Pct.  | Cham. | Wins  | Loss. | Scor. | Agai. |
------------------+-------+-------+-------+-------+-------+-------|
Real Madrid       | 1.000 |     1 |     1 |     0 |     3 |     2 |
Manchester United | 0.500 |     1 |     1 |     1 |     3 |     3 |
Barcelona         | 0.000 |     0 |     0 |     1 |     0 |     1 |
------------------+-------+-------+-------+-------+-------+-------|
TOTALS            |       |     2 |     2 |     2 |     6 |     6 |
```

The size of each column in the output is:

```
<--    n      --->|123456 |123456 |123456 |123456 |123456 |123456 |
Barcelona         | 0.000 |     0 |     0 |     1 |     0 |     1 |
```

where n is the length of the largest team name. The columns are: Pct is the number of wins, divided by number of matches played. $Wins$ is the number of won matches, $Loss.$ is the number of lost matches. $Scor.$ is the number of scored points in all matches, $Agai.$ is the number of points received in all matches. The Pct column should include 3 decimals (no negative numbers are needed). You can safely assume the rest of the number will fit in the 6 spaces allocated for them. All numeric values should be aligned to the right. No TAB character should be used in the output.

If two teams have the same percentage (as computed internally, not as printed) the output should be ordered using the following rules (used recursively):

- The tied teams are ordered by number of won championships, in descending order.
- Otherwise, the tied teams are ordered by number of points difference (points scored minus points received), in descending order.
- Otherwise, the tied teams are ordered by number of scored points, in descending order.
- Otherwise, the tied teams are listed in ascending lexicographical order (as defined by `strcmp`).

Requirements Notes

- If your program does not find enough memory to continue, it should issue an error message and immediately exit with error code 1.
- You can assume that no team has a name longer than 1024 characters long (not including NULL character).
- You **should not modify** simple.c in the current implementation. You can modify any other files, and you can add new files. I will provide my own **simple.c** during the testing of your implementation.
- If your modifications require new functions, you are free to write them.
- If you add new files, the Makefile should be modified accordingly.
- Every time you read a new tournament (Task 3), your program should release the memory that it no longer needs (for instance, if the same 4 teams play 1 Giga tournaments, your program **should** not run out of memory.
- And when your program terminates properly, it should first release **all the memory** it has used.

Implementation Notes

- You will find my solution in the directory `~seng265/assign4`
- You are required to use the `dmalloc` library (as described in the lab) for debugging purposes. I will use it to test your assignment.
- You should **obbey** the coding standards.

What to submit

Submit all the files via CVS, using the module of one of your team members, in a directory called assign4.

1. Create a file called `INFO` that includes for each member in the team, a line like follows:

   ```
   userid:studentid:Name
   ```

Finally, at the start of the class on the deadline, **hand in a printed copy of your `cvs commit` script as well as a hard copy of your all the files you modified**.

Remember, no line in your source code should exceed 80 characters long!