CSCI-1200 Data Structures — Fall 2009
Homework 2 — Decathlon Classes

In this assignment you will parse and compute statistics from the results of the Olympics track and field decathlon competition. Scoring and determining the winner of this competition is a bit involved, but we have summarized everything you need to know to complete this assignment. Please read the entire handout before starting to code the assignment.

The Decathlon

Here's a crash course in scoring: Each decathlete competes in a total of 10 specific events over the course of 2 days. The 4 running events (100 meters, 400 meters, 110 meters hurdles, and 1500 meters) are *track events* and measured in seconds (faster times are better). The other 6 events (long jump, shot put, high jump, discus throw, pole vault, and javelin throw) are *field events* and measured in meters (longer distances are better).

To determine the winner of the entire competition, the scores for each event are converted to points using the formulas and constants below and added together. The formulas have been repeatedly adjusted over the years to normalize the units for the different events and appropriately reward typical and outstanding performances in each event. For more information see `http://en.wikipedia.org/wiki/Decathlon`.

For track events:

$$\text{points} = \text{int}(A * (B - \text{score})^C)$$

For field events:

$$\text{points} = \text{int}(A * (\text{score} - B)^C)$$

Event	A	B	C	Units
100m	25.4347	18	1.81	seconds
Long Jump	0.14354	220	1.4	centimeters
Shot Put	51.39	1.5	1.05	meters
High Jump	0.8465	75	1.42	centimeters
400 m	1.53775	82	1.81	seconds
110 m Hurdles	5.74352	28.5	1.92	seconds
Discus Throw	12.91	4	1.1	meters
Pole Vault	0.2797	100	1.35	centimeters
Javelin Throw	10.14	7	1.08	meters
1500 m	0.03768	480	1.85	seconds

You will parse an input file with the scores for each event (not in any particular order). Here is a partial input file:

```
event POLE_VAULT
Andrei Krauchanka BLR 5.00
Bryan Clay USA 5.00
Leonel Suarez CUB 4.70

event 100_METERS
Bryan Clay USA 10.44
Leonel Suarez CUB 10.90
Andrei Krauchanka BLR 10.96

event 1500_METERS
Andrei Krauchanka BLR 4:27.47
Leonel Suarez CUB 4:29.17
Bryan Clay USA 5:06.59
```

The scores for each event are delimited by the keyword `event` followed by the name of the event. There are no spaces in the event name. Then each player is listed with first name, last name, country, and score for the event. All of the scores are given as a floating point number in the units (seconds or meters)

appropriate for the event. The exception is the 1500 meter run, which is given in the format `m:ss.ss`. Some decathletes are not listed for each event because they did not complete the event or because their performance was disqualified and thus they will receive no points for that event.

File I/O and Command Line Arguments

Your program will run with two command-line arguments, one being the name of the input file containing the raw event scores and the other being the name of the output file where you will write the computed statistics. Example input and output files are posted on the course website. For example, here is a valid command line to your program:

```
decathlon_statistics.exe 2008_medalists.txt out_2008_medalists.txt
```

We have provided you with decathlon score datasets from the 2000, 2004, and 2008 Olympics. The original data was taken from the International Association of Athletics Federation's website, `http://www.iaaf.org/`. The format has been modified to ease parsing.

Statistics Collected and Output

The output will be in *three parts*. First is a table with the decathletes sorted alphabetically by country abbreviation, then by last name. Each row of the table should include the decathlete's first and last names, country name, and the scores for the 10 events. For example, given an input file with scores for the 3 medalists from the 2008 Olympics, your program will produce a table similar to this:

DECATHLETE SCORES			100	LJ	SP	HJ	400	110H	DT	PV	JT	1500
Andrei	Krauchanka	BLR	10.96	7.61	14.39	2.11	47.30	14.21	44.58	5.00	60.23	4:27.47
Leonel	Suarez	CUB	10.90	7.33	14.49	2.05	47.91	14.15	44.45	4.70	73.98	4:29.17
Bryan	Clay	USA	10.44	7.78	16.27	1.99	48.92	13.93	53.79	5.00	70.97	5:06.59

The formatting shown above is an example; your output may be formatted differently as long as it is easy to read. If the athlete did not score a particular event, that cell in the table should be blank. The second part of the output is a table where the players are sorted by their total summed points for the competition. Again each row includes the decathlete's first and last names and country name. The points earned for each event are listed, followed by the total. The formatting shown below is an example of how to present this information:

DECATHLETE POINTS			100	LJ	SP	HJ	400	110H	DT	PV	JT	1500	TOTAL
Bryan	Clay	USA	989	1005	868	794	865	984	950	910	904	522	8791
Andrei	Krauchanka	BLR	870	962	752	906	943	948	758	910	741	761	8551
Leonel	Suarez	CUB	883	893	758	850	913	955	756	819	950	750	8527

The third and final part of the output is a chance for you to be creative. Brainstorm a new interesting statistic that can be calculated from this data. Examples include determining and outputting how many of the 10 events each decathlete "won", placed "second", placed "third", etc. Another example would be to compute the average and standard deviation of points earned for each event which could indicate which events have more of an impact on the competition winner.

Extra credit will be awarded to particularly interesting statistics that require clever programming. The most important task for this part of the assignment is to write a concise description (< 100 words) of your new statistic. Put this description in your *plaintext* `README.txt` file along with any other notes for the grader. Be sure to tell the grader which dataset best demonstrates your new statistic. Feel free to create your own dataset and include it with your submission.

Useful Code

To control the formatting of your tables, you'll want to read up on the various iomanipulators: `std::setw(int)`, `std::setprecision(int)`, `std::fixed`, `std::left`, etc. And don't forget about the `sort` function that can be used to order the contents of a `vector`.

Program Requirements & Submission Details

Your program should involve the definition of *at least one class* that has its own .h and .cpp files, named appropriately. Initially you should focus on the small dataset with the 2008 Olympic medalists. Once that is working you can extend your solution to handle the bigger test cases which include data for decathletes who did not complete all of the events.

Do all of your work in a folder named `hw2` inside of your Data Structures homeworks directory. Use good coding style when you design and implement your program. Be sure to make up new test cases and don't forget to comment your code! Please use the provided template `README.txt` file for any notes you want the grader to read. **You must do this assignment on your own, as described in the "Academic Integrity for Homework" handout. If you did discuss the problem or error messages, etc. with anyone, please list their names in your `README.txt` file.** When you've finished writing, testing, debugging, and commenting your code, prepare and submit your assignment as instructed on the course webpage. Please ask a TA if you need help preparing your assignment for submission or if you have difficulty writing portable code.