CSCI-1200 Data Structures — Fall 2016
Homework 2 — Swimming Classes

In this assignment you will parse and compute statistics from the results of international swimming competition, specifically the recent Summer Olympics in Rio.

Here's a crash course in swimming: There are four basic strokes: freestyle, backstroke, breaststroke, and butterfly. A standard Olympic pool is 50m in length and can accommodate 8 lanes, with one swimmer per lane. There are 32 different events in a modern Olympic swimming meet, 16 for women and 16 for men.

https://en.wikipedia.org/wiki/Swimming_at_the_Summer_Olympics

The different competition events are different distances and have different prescribed strokes. Three of the women's events and three of the men's events are "relay" races, that bring together four swimmers from the same country and they each swim one quarter of the total distance for that race. Some of the events are "medley" events that require the swimmer (or team of four swimmers) to use each stroke for a specified quarter of the race.

Because generally more than 8 people or teams will compete in an event, the event is broken into two or three stages: "heats" and "finals", or "heats", "semifinals", and "finals". Swimmers much achieve one of the top scores in their "heat" or "semifinal" (depending on the number of participants) to progress to the next round. The medals are awarded to the top performers in the final round of the event (results from earlier rounds do not count).

Weird Factoids about Swimming Competition

Even though modern technology allows us to measure time differences extremely accurately, international swimming results are only recorded to 1/100th of a second. Why? Because elite swimmers at their top speed can cover almost 3 cm in 1/100th of a second. And 3 cm is the allowable construction tolerance for the Olympic-compliant pools. In other words, it is not worth measuring more accurately because differences in the length of two different lanes of the pool would potentially decide the outcome of the race. Therefore, ties are not unusual in swimming competition.

http://regressing.deadspin.com/this-is-why-there-are-so-many-ties-in-swimming-1785234795

With access to data and visualization tools, analysis of the results lap split times of competitors at the Rio Olympics (and earlier international competitions including prior Olympics) has led to speculation that some pools have a subtle but measurable overall circular current that is pushing against the swimmers in the low numbered lanes as they swim away from the starting blocks (slowing them down) and helping them along as they return to the starting blocks (speeding them up). In events with an even number of laps, the effect of this current is probably canceled out. But for the shortest race with a single length of the pool (no return), this could be a noticeable advantage or disadvantage (depending on the swimmer's lane assignment).

https://swimswam.com/problem-rio-pool/

Since we have the full Olympic dataset to play with for this assignment, you are welcome and encouraged to do your own analysis and draw your own conclusions.

Sample Data

Here's a small portion of the data for the final round of the Women's 200m Freestyle. Each event begins with a keyword "EVENT" followed the the full title of the event.

```
EVENT
Women's 200m Freestyle
FINAL    3  Federica PELLEGRINI          ITA     27.09     56.45   1:25.84   1:55.18
FINAL    4  Sarah SJOSTROM               SWE     26.84     55.86   1:24.95   1:54.08
FINAL    5  Katie LEDECKY                USA     27.00     55.43   1:24.55   1:53.73
FINAL    6  Duo SHEN                     CHN     27.07     55.90   1:25.67   1:55.25
```

The input data comes from https://www.rio2016.com/en/swimming-featured-results-download. The format has been modified to ease parsing.

Each participant result from the event begins with a keyword indicating which round of competition it is from. Note: Participants in the final of this event also have results from the one of two semifinals and one of six heats. The heat and semifinal results have an extra integer column indicating which semifinal or which heat this result came from. Next is the lane assignment. It is generally accepted that the middle lanes are advantageous, and thus are awarded to the highest finishers from the previous rounds.

Each swimmer is identified by a first and last name. The data has been cleaned up a bit for you. Swimmers with multiple first names or multiple last names have been combined with an underscore to allow easy parsing. Next is the three letter country code. Finally, this dataset includes the intermediate timing results for every 50m length of the pool. The final number is the swimmer's total time to complete the distance for this race. The format is minutes + seconds to the nearest 1/100th (m:ss.ss).

The team relay events are formatted similarly, but instead of a 2 part first and last name, there is just one string. Multi-word country names are combined with the underscore character. And on a line below the team's timing results are the names of the four swimmers (in order) who participated in that race.

File I/O and Command Line Arguments

Your task is to write a program to read, manipulate, and output this data. Your program will accept 3 or 4 command-line arguments. The first is the name of the input file, the second is the name of the output file. The third controls what type of data should be output. And the fourth argument is optional, indicating that the basic table requested should be expanded with more information.

For example, here are valid command lines to your program:

```
./swimming_statistics.out  sample_dataset.txt  sample_output_events.txt               events
./swimming_statistics.out  sample_dataset.txt  sample_output_events_medals.txt        events        medals
./swimming_statistics.out  sample_dataset.txt  sample_output_participants.txt         participants
./swimming_statistics.out  sample_dataset.txt  sample_output_participants_medals.txt  participants  medals
./swimming_statistics.out  sample_dataset.txt  sample_output_custom.txt               custom
```

Statistics Collected and Output

Depending on the third command line argument ("`events`", "`participants`", or "`custom`") your program will output one of three different types of tables. The optional fourth command line argument "`medals`" is a bit more involved. The optional command line argument will be worth a smaller number of points, so we *strongly* recommend that you complete the basic tables first. A full credit solution must be able to produce both the basic and extended version of each table. First, let's look at the simple "`events`" table:

```
Women's 200m Freestyle                  50m     100m     150m     200m       FINAL
-------------------------------------------------------------------------------
Katie LEDECKY                         27.00    28.43    29.12    29.18     1:53.73
Sarah SJOSTROM                        26.84    29.02    29.09    29.13     1:54.08
Sarah SJOSTROM                        27.07    29.16    29.42    29.00     1:54.65
Katie LEDECKY                         27.20    29.01    29.57    29.03     1:54.81
Emma McKEON                           26.64    28.73    29.80    29.75     1:54.92

Women's 400m Freestyle                  50m     100m     150m     200m     250m     300m     350m     400m       FINAL
-------------------------------------------------------------------------------------------------------------------------
Katie LEDECKY                         27.73    29.32    29.94    30.12    30.30    30.21    29.92    28.92     3:56.46
Katie LEDECKY                         28.24    29.96    30.26    30.48    29.81    30.36    29.99    29.61     3:58.71
Jazz CARLIN                           28.49    30.37    30.39    30.83    29.99    30.61    30.48    30.07     4:01.23
Leah SMITH                            28.42    30.47    30.70    30.93    30.58    30.60    30.29    29.93     4:01.92
Boglarka KAPAS                        28.88    30.79    30.28    30.73    30.69    30.95    30.39    29.66     4:02.37
```

For each event in the input file we create a short table with the lap splits (difference between timing measurements) and the total results. You'll need to calculate the splits from the intermediate timing data in

the file. This table allows a coach to study how their swimmer's pace is varying over the course of the race and compare it to the other successful racers. This table only shows the top 5 results in this event, sorted by completion time. If there is a tie in total time, the swimmers should be sorted by last name, then first name. Note there might be duplicate names in this table, if a swimmer had top scores in multiple rounds of competition. The tables are ordered in the output file alphabetically by the full event name.

In the "`medals`" version of this table, the gold, silver, and bronze medalists are labeled with an extra column on the right. Note: This is not always the first 3 rows and with great results in early rounds and ties, the bronze medal might not be in the top 5! Instead we show at least 5 results and all results through and including the first non medalist in the final round. We also label the non final round results next to the swimmer name.

```
COUNTRY  PARTICIPANT          HEATS  SEMIS  FINALS
-------------------------------------------------
AUS      Bronte BARRATT          0      1       1
                   <snip>
SLO      Anja KLINAR             1      0       0
SRB      Katarina SIMONOVIC      1      0       0
SWE      Michelle COLEMAN        0      1       1
SWE      Sarah SJOSTROM          0      1       1
USA      Missy FRANKLIN          0      1       0
USA      Katie LEDECKY           1      1       2
USA      Leah SMITH              1      0       1
VEN      Andreina PINTO          1      0       0
VIE      Vien NGUYEN_THI_ANH     1      0       0
```

Next we discuss a subset of the simple "`participants`" table. This table contains all swimmers for all events in the input file. The swimmers are sorted first by country abbreviation, then by last name, then first name. The remaining columns in this table store counts of the different rounds this swimmer participated in. In the "`medals`" version of this table, we add three columns to count the swimmers' medals. Additionally, to make the table more legible we use a blank instead of a zero to highlight the swimmer's success. When the relay events are included, we account for each swimmer's participation in those events as well. But we do not list the relay team countries in the participants table.

The third and final part of the output (when "`custom`" is specified on the command line) is a chance for you to be creative. You will collect and output some other statistic from the matches. We encourage you to explore performance in the four different strokes, including during *medley* events. Or as mentioned earlier you could study the outgoing and incoming split time differential for the different lanes. Or you could create a fantasy swimming league mixing up the countries to which each swimmer is assigned.

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 `README.txt` along with any other notes for the grader. Be sure to tell the grader which dataset best demonstrates your new statistic. Create your own dataset and include it and your program's output for that test case with your submission.

Program Requirements & Submission Details & Useful Code

Your program should involve the definition of *at least one class* that has its own `.h` and `.cpp` files, named appropriately. As with Homework 1, you should also make good use of the STL `string` and STL `vector` classes. You do not need pointers, the `new` operator, or recursion for this homework.

You should aim to match the sample tables nearly exactly. *Hint: use the* `diff` *command from your terminal to check your output vs. the sample output.* To control the formatting of your tables, you'll want to read up on the various iomanipulators: `std::setw(int)`, `std::setprecision(int)`, and `std::fixed`, using `#include <iomanip>`. And don't forget about the STL `sort` function that can be used to order the contents of a `vector`. We recommend you use the `>>` input stream operator to fully parse the input file. You do not need the `getline` or `eof` functions. See also "Misc. C++ Programming Information".

Be sure to read the "Homework Grading Criteria" as you put the finishing touches on your solution. Use the provided template `README.txt` file for notes you want the grader to read. You must do this assignment on your own, as described in the "Collaboration Policy & Academic Integrity" handout. If you did discuss this assignment, problem solving techniques, or error messages, etc. with anyone, please list their names in your `README.txt` file.