# Running laps:

In this lab you will be reading in a file of student results. The students each ran 3 laps in a race and their times to complete each lap are posted in the order that they completed the lap (the students will not necessarily be in the same order each lap). You will be outputting a number of results based on the student performance. And it should go without saying that you cannot hard-code any of the values into your program as the values I’ve given as an example are not the same values I will use to test it with. There is a maximum of 20 students that may participate in the race (though the example file only has 7).

Hint: times in the files are given as minutes:seconds. But many times, you need to add or do calculations with them together. So you will need to convert them to total seconds to do these calculations. Then for displaying you will be converting them back.

Hint #2: Do **not** store the values in your program. Many students get advice on how to do this from *other sources* and whereas this could have value in many situations (other labs maybe) here it is completely unnecessary. The text file stores the values already, so there is no need to use arrays or lists or anything of that kind to do what the text file already does. If you need to compare one value to another you can read the file multiple times.   
tldr: Yes, reading a file can be slower than accessing it from an array or list but not overly so. The only real slow down with using streams is that the stream could be on another source machine over a network for instance – in that case an array to store the values is very much preferred. But here, we are using a file on a local machine (that will get cached anyways), so rereading the file x amount of times isn’t really a problem.

Objective 1: Output the final times of all the students. I also want to know who placed 1st, 2nd and 3rd overall. (Though if you have them all in order that will be sufficient). You can order values simply by setting aside 3 variables and when you calculate a new score you see if it is the best, if so move the old best into 2nd place and move the 2nd place into 3rd place. If it isn’t the best, then move down to 2nd and try that one and so on.

I should also note that you may read a file as many times as you want. Though it is not necessarily the most efficient solution you may read the file over again for each student that participated.

Objective 2: I want the averages for each lap by all the students. Then output which students are above the average and which are below:

Lap 1 average: 2:05  
Below: Akano, Wes, Kye, Edward (note that Edward is right on the border and could be put in either)  
Above: Jess, Ally, Wilt

Objective 3: Naturally, the students slowed down from lap to lap as they were running. I want the lap times and the difference between them posted for each student:

Lap 1 2 3  
Akano 1:48 2:28 2:25  
 +40 -3 (note that she is one of the few that needs a negative)

Objective 4: Consistency in races is important. I want to know the total range of each students fastest and slowest lap. In the end I want to know the top 3 most consistent runners:

Slowest fastest difference  
Akano: 2:28 1:48 40 sec

Objective 5: Now you are going to use both the example files together. The second results file contains the same students (though my test data will be 2 files with different number and names than the files you are given). I want a comparison of the student’s overall times from each results file:

1 2 difference  
Akano: 6:41 5:49 -52 sec