Task 1: Hello

This is a simple problem to read in a single string from the input and print out and hello message for that string. This problem is set to let students try out reading from standard input and writing to standard output in all 3 languages: C++, Java and Python.

Task 2: Sum

This problem requires us to read N pairs of integers X_i, Y_i and print the sum of X_i and sum of Y_i. This problem should be simple enough to be solved by anyone who has learnt about loops in programming. However, the difficult part of this problem is to read in the input within the time limit as N can be up to 10^7. The input of this problem actually matches that of the problem Lightning Rod in the actual contest so it helps students to prepare for that.

As we do not expect students to know about such speedups, we have provided templates to help students read the input and focus on the processing. In C++, using *cin, cout* will result in Time Limit Exceeded. It is recommended that *scanf, printf* are used instead. For Java and Python, most ways of input and output will result in Time Limit Exceeded, thus it is recommended that the input be buffered and then the buffer be processed separately. However, Python is simply too slow and the provided template will only reach subtask 2.

Task 3: Sorting

This problem requires us to sort a list of numbers, but without access to the list. We can query the grader to compare two of the numbers but we only have a limited number of queries. This problem is meant to introduce students to the Interactive task format as there was a similar problem in the actual contest.

For subtask 1, there are only 2 numbers so we only need 1 query to compare these 2 numbers and we can decide their order. For subtask 2, we can perform any $O(N^2)$ sort such as bubble sort and when comparing two numbers, we simply call the query function instead of accessing our own array.

For subtask 3, we can use the *sort* function in the *algorithm* C++ standard template library and change the comparison function to the query function instead. However, if we try, we will notice that we do not pass subtask 1 as sometimes it would use 2 comparisons to compare 2 numbers due to the way it is implemented. There are 2 ways to get around this. One is to check if $N = 2$ and if so, we implement our solution from subtask 1, otherwise we use the *sort* function. Alternatively, we can replace the *sort* function with *stable_sort* and we will also pass the first subtask. The reason why is left for the reader to explore and involves how the functions are implemented inside the library.