1. Preamble

The Hogwarts Quidditch Cup will kick off in November 2022. The tournament starts with eight groups of four teams. Every team plays three matches with other three teams in a group. The group stage rules and tiebreakers below determine the final group rankings table:

• The first thing to look at is the total point each team gains from the matches. Three points for a win, one point for a draw, zero point for a loss.

• If teams are equal on points, Hogwarts uses the following Quidditch Cup tiebreakers. First, the goal difference is considered. Goal difference is the total of goals a team scores in three matches played, subtracted by the goals they let in.

• Next thing to consider are the goals scored. Higher the number of goals scored in three matches, the better.

After group stage rules and tiebreakers are applied, based on the total points each team achieves, the rankings will tell which top two teams advance to the knockout stage of the tournament, while the other two teams pack their bags and board a plane back home.

2. Assessment Task

Design, construct and test a C program which, without any prompt message, reads keyboard input, two integer numbers per line to record the scores of all the matches between the teams. After six lines of scores are input, using only the first three rules above, your program ranks and prints the order of four teams in exact the example format shown below.

If four capital letters, A B C and D, represent four teams in the group for ranking. The input order and scores of the matches should exactly follow the match sequence as A:B, A:C, A:D, B:C, B:D and C:D. Please note the scores are only integers. An example input from the keyboard to your code is shown below:

```
10
20
32
22
30
00
```

Output of your code should produce such output: **`Ranking of the teams is ABCD.`**

3. Observations on the Task

After the first three stage rules and tiebreakers are applied, if two or more teams still have the same scores, you just rank them using an alphabetical order.

Proper use of functions, along with testing of each function individually before integration into the complete program, will assist in ensuring correct operation, as well as giving a readable and maintainable program with a clear modular structure. Such an approach will make it possible to create a simplified version of the program early on.

Consider carefully all the possibilities in the matches. After the first three stage rules and tiebreakers are applied, if two or more teams still have the same scores, you just rank them using an alphabetical order.

You may expect to produce several revisions of your program, some of which include intermediate output used only for your diagnostic testing purposes. Do not leave active instances of such output in the submitted version of your code.

Proper use of functions, along with testing of each function individually before integration into the complete program, will assist in ensuring correct operation, as well as giving a readable and maintainable program with a clear modular structure. Such an approach will make it possible to create a simplified version of the program early on.

When testing your program, you should prepare a number of different match scores with different possible arrangement of win, lose and tie as test input, then compare the output of your program with the expected correct output. You should comment on your code throughout to explain the meaning of the key instructions. You should use as many knowledge taught in the first four C language sessions as possible, such as pointer and function. You are encouraged to use knowledge in the 5th C language session, but it is not compulsory. (**5th session = bitwise operations, dynamic memory allocation**).

~~~

You may choose to develop your program on a platform rather than the usual es2cc we use in the lab. In that case, you must check, before submission, that your program behaves as you expect when compiled and executed using the es2cc complier. The correctness will be assessed on the basis of compilation and execution, in addition to inspection of the source code.
~~~