Coding Challenge

Solving programming challenges is a fundamental skill within computer science. When developing complex programs there are often many challenges to solve both when implementing features or when performing the various other tasks required for successful software development. In the real world, you cannot be expected to know how to do everything off the top of your head, therefore the ability to efficiently research is also a key skill in being an effective software developer.

Context

As we all know sailing is one of the most historic and best sports to exist. Something equally as prestigious is the Olympic games which takes place every 4 years. Next year this happens in Paris, France but the organisers need help calculating the score for the sailing. This is your job!
Below is a series of tasks for you to complete, each task somewhat builds on from the last and vary in difficulty. Here you will solve the tasks within python by applying the skills you have learned within this course unit and some individual research.

In our scenario there are a total of 10 countries participating in the Olympic sailing event. Each country is given a unique number from 01 to 10. There are also 10 types of boats being raced, again each being given a unique number from 01 to 10. Each country is represented exactly once in each type of boat. There are two different types of races. Race 01 is a single points race and Race 02 is double points. Points are awarded to the sailor's respective placing in a particular race, e.g. if a boat finishes 1^{st} then they receive 1 point, 2^{nd} gets 2pts and so on until 10^{th} gets 10pts. The winner of a racing series is that with the lowest points total.

Input File Structure

```
0201-0701-0602-0503-0104-0905-0306-0807-0408-0209-0210
0602-0201-0402-0603-0504-0905-0306-0707-0108-0809-1010
0501-0201-0702-01xx-0804-1005-0306-0907-0508-0409-0610
0701-0301-0902-0103-0604-0205-0406-0707-0508-1009-0810
```

The first 4 digits (i.e. `0201`) represent the Type of Boat and the Type of race taking place. So `0201` mean Boat Type 02 and Single Points Race. `0602` would mean Boat Type 06 and double points race.

There may be an occurrence where you see a boats finishing position is xx such as boat 01 below:
`0501-0201-0702-01xx-0804-1005-0306-0907-0508-0409-0610`

This means that this boat has been disqualified from the race and will receive points totalling the total number of boats + 1 (i.e. 11). Every boat which finished behind this will now also move up one place. i.e. boat 08 will now finished 3^{rd} instead of 4^{th}, boat 10 4^{th} in stead of 5^{th}, etc. If there are multiple boats disqualified from a race they will all recieve 11 points irrespective of their original position in the race.

Tasks

In this work, there are 6 tasks to complete. Each of these tasks is associated with a method within the provided .py file however there are dependencies within this work. These tasks are each part of a process which should be followed in order, examples of expected output formatting are given later in this document.

Section 1: File Input

(a) Read and store all of the race results into a list of strings which for this specification we will call the "results string"

(b) Each race in the results.txt file is on a new line

(c) Return the results string

Section 2: Specific Race Results

(a) Output the results of Boat 'n' race 'x' where n & x are give by the automarker

(b) the race number for a boat is the order in which they are read from the input file. i.e. the first occurrence of boat type 01 is their first race, the second occurrence is their second race, and so on.

(c) Return the race results string

Section 3: Boat Type Results Table

(a) Output the overall results for class 'x'. x is provided by the automarker

(b) Output format is "CountryNumber-PositionInTable-TotalPoints" e.g. `05-01-25`

(c) These should appear in accessing order with the winning boat (i.e. lowest points total) appearing first

(d) If there is a tie on points the boat that performed best in the last race is ahead on the leaderboard

(e) Return the class table string

Section 4: Results Tables inc. Discards

(a) Each individual boat can Discard their worst 2 results and exclude them from their total score

(b) Each racer will discard their worst single point race and their worst double points race

(c) If a sailor has multiple races with the same worst score remove the last instance of it

(d) A sailor discarding a result does not effect the results of other sailors in that race

(e) If there are 2 or less races of a certain type then all sailors count those scores and no discard is applied to that race type for that boat

(f) Return the class table discard string

Section 5: Overall Country Medal Table

(a) Determine how many medals each country wins. If a boat finishes in the top 3 of a certain boat type then they will the respective medal for that boat

(b) Positionings are based on the results tables from section 4

(c) A Gold meal is worth 3pts in the Medal table, Silver = 2pts, Bronze = 1pt

(d) Output format is "Country-Golds-Silvers-Bronzes-TotalMedalsScore" e.g. `03-02-04-01-15` or `05-00-01-00-02`

(e) Display order is total medals score, then golds, silvers, bronze, country number

(f) Return the medal table string

Section 6: The Country & Class with the most Disqualifications

(a) Determine the naughtiest Country and Boats from the event

(b) Output format is "ContryNumber&SumOfDisqualification-BoatTypeNumber&SumOfDisqualification". e.g. `0307-0704` would represent country 03 has a total of 07 disqualifications across all of the races and Boat type 07 has the most disqualifications.

(c) Return the disqualifications string

Sample Outputs for Each Task

Below are a selection of example outputs for each task, these do not necessarily correspond with each other and are a formatting guide only.

- Section 1: [`'0902-0701-0302-0403-08xx-0605-1006-0207-0508-0909-0910'`, `'0201-0701-0602-0503-0104-0905-0306-0807-0408-0209-0210'`, `'0402-0801-0902-1003-0604-0305-0406-0107-0208-0509-0510'`, `'0401-0601-1002-0703-0104-0405-0806-0307-0208-0909-0910'`, `'0101-1001-0802-0403-0704-0105-0606-0307-0908-0209-0210'`]
- Section 2: "`0601-0902-1003-0304-0505-0106-0707-0808-0409-0210`"
- Section 3: "`05-01-25, 03-02-36, 06-03-41, 08-04-43, 04-05-44, 09-06-45, 07-07-48, 10-08-50, 01-09-53, 02-10-58`"
- Section 4: "`05-01-21, 03-02-27, 06-03-33, 04-04-34, 08-05-36, 09-06-37, 07-07-38, 10-08-39, 02-09-47, 01-10-47`"
- Section 5: "`06-01-03-02-11, 03-01-03-00-09, 05-02-01-00-08, 09-02-00-01-07, 08-01-01-02-07, 07-02-00-00-06, 01-00-01-01-03, 10-00-00-02-02, 04-00-00-01-01, 02-00-00-00-00`"
- Section 6: "`1002-0602`"

Provided Materials

For this work we provide you with two files:

- A Python file called `sailing_calculator.py`. This will need to be renamed into the following format:
  [user_name]_midterm.py (*e.g. a12345bc_midterm.py*)
- An example test file called `input.txt`

Important Notes:

1. The input file should always be in the same directory as the python file *(and not contained in any other folder)*
2. All instructions within the Python file should be followed exactly
3. The example test file given will not be used for marking purposes
4. The text file will always be named "input.txt"
5. Import statements are **not** allowed for this task
6. There is no required output for this work. All output is done via method return statements. Methods will be called directly. Print statements may cause the tests to fail to so avoid them or remove them before submission.
7. You are only permitted to submit a single python file for this work (do not submit test files or any other files)
8. You do not need to submit the 11input.txt" file
9. Do no you use absolute file paths or the auto-marking will fail with **no** exceptions for this

Preparation and Submission

Your [*user_name*]_midterm.py file must be pushed to your Gitlab (*COMP16321-Labs_[username]*), it must have the tag *Midterm-Test* and be on a branch called *Midterm*. Failure to correctly tag your work or placing it on an incorrect branch may cause the marking to fail resulting in a mark of 0.

Your code will be run through an automated marking system, Codegrade. Making amendments to the file you were provided may cause the marking to fail and you will not receive the marks for the effected portion.

Deadline:

18:00 on Friday the 9th December 2022 (*Note, there are no DASS extensions for this work*)

Assessment Type:

This activity is subject to **summative** assessments regulations, therefore your submission will be marked and you will receive the associated feedback. The marks you obtain (see below) count for up to 10% of your overall mark for this course unit.

If the marker finds that the file submitted is unreadable and/or does not compile then you will not receive any marks for that program. Please be aware that we will not install external packages to make your code compile. We should be able to run your code with a basic python setup and the packages outlined at the start of this document.

Please note: Your work will not be re-marked because you disagree with the mark you were awarded. The markers all follow the same rubric which has been designed to be as fair and comprehensive as possible. Any queries with your marking can be raised with either Gareth or Stewart within a week of the marks being released, any queries made after this time will not be considered.

End of Assessment