**Design and Analysis of Data Structures and Algorithms.**

**Coursework B Specification – Academic season 2017-18**

**Coursework Submission 22 March 2018 at 2pm**

**Coursework contributes 70% to the overall coursework marks.**

**Scenario**

The same tournament system used for coursework A is required for this coursework too.

There are four tournaments to be played in any season namely,

There are both Men's and Women's tracks for each tournament.

There are 32 players for each of the two tracks.

The principles are the same a male player needs to achieve 3 sets to win a match while a female player needs to win two sets to win a match.

No draws are allowed in a tennis match and if such a score is encountered the system should request the score to be inputted again. If while reading from a file an erroneous score (a draw) has been encountered, the system should process the rest of the match scores in the file and ask the user to input the correct score manually.

For this coursework we will be accepting that a player can withdraw at any point in the match (due to injury). In that case a win of 3 sets to 2 sets (men) or 2 sets to 1 (women) will be recorded to the credit of the non-withdrawing player. One way for handling this is that if an incomplete score is encountered the user will be asked to verify if the score is valid due to a player having withdrawn due to injury. If such is the case, the withdrawn player will be marked as a loser for this match.

The system will have to allow a user to switch between results entry modes between rounds of a tournament.

Actual scores have to be processed for each match. The scores will determine the number of points won by a player.

If a male player wins by 3 sets to 0 then the base points awarded for that round must be multiplied by a factor of 2.5

If a male player wins by 3 sets to 1 then the based points awarded for that round must be multiplied by a factor of 1.5

There is no bonus factor for a 3 sets to 2 win.

In the ladies tournaments, a win by 2 sets to 0 will yield a bonus factor of 2.5 by which the base points awarded for that round must be multiplied.

The software must save all scores in a repository, to allow for wins or losses for an individual player to be processed. For example the system might be required to produce the number of wins by a score of 3 sets to 1 that have been achieved by a specific player at that point in time in the season.

**Data Provided**

The scores for all matches, for all rounds, for all tournaments are given to you in CSV files.

The player lists are also given.

Base points for player rating are also provided.

The tournaments' degree of difficulty and the score bonus factors are also given.

Finally, the prize money per round is provided.

**Tasks to be completed**

Task 1 – Design a solution (produce relevant pseudo code) to allow the processing of scores for both input types. At the end of each round the user interface will have to provide the user with the winners of that round and the options to go to the next round or exit. On choosing the next round the user will have to select whether they want to read results from files or to enter them manually. Once all rounds have been completed the winner will be declared and the user will be offered the choice to see the tournament's ranking (points or prize money), the overall season ranking, proceed to the next tournament, or exit the program. At any point of time all processed data must be saved upon exit from the program.

Task 2 – Implement the solution designed in Task 1 in Python.

Task 3 – Design (pseudo code) relevant algorithms that will allow you to show the user the following.

- The number of wins for a player with a particular score – either in a specific tournament, or overall for the season.
- The percentage wins of a player in a specific tournament or for a whole season.
- Show the player with most wins in the season so far.
- Show the player with most loses in the season so far.

Task 4 – Implement the design shown in Task 3.

Task 5 – Evaluate the efficiency of your software, in terms of the size of the code, speed of running and efficient use of functions and specialist algorithms. You can implement an evaluation algorithm and you can discuss the key features of your work in design and implementation.

Task 6 - A second season will be introduced – player pairings will be governed by player seedings. The final placements (based on ranking points earned) from the previous season will be used to create the first round pairs for each tournament. The first sixteen in a season cannot play each other in the first round of the tournament. Once in the second round of a tournament, player seedings will be based on the relevant positions achieved in the same tournament during the previous season. So those that have played in the last eight in the previous season of the same tournament cannot face each other in the round of the last sixteen players in the new season, those that have reached the semi-finals in the previous season cannot play each other in the last eight round and finally those that have reached the final in the previous season cannot play each other in the semi-finals.
The points' difficulty factor (for the tournament) will only be applied if a player has at least achieved the same position as per the previous season in the same tournament.
At the end of each tournament ranking (points and prize money) will be made available to the user as choices to see. These will include ranking for the tournament, the current season, or combining those of the previous season with the current season.
Design the addition to the software to provide this functionality (pseudo code).

Task 7 – Implement the design for Task 6.

**Assessment Criteria / Mark Scheme**

Full marks for each section below will be given for fully working code.

Other requirements are:

- Clear comments within the code that link well with the pseudocode provided
- Pseudocode must simple but effective. Covering all functionality of your software solution using structured English – no machine language like code.
- Justification; the text needs to be simple, well structured, providing clear comprehension in a non-verbose way
- Efficiency of the design and code produced will be required to reach the top end of the marks available for a set of tasks – e.g. good use of functions, minimal use of repetitive code, etc.
- Hard Coding of scores, player rankings and player earnings is not acceptable. Any submission that contains such hard coded data (whether intentionally or "by accident") will be marked at 0 (zero)!

To achieve a **PASS (40%)** you need to complete the following tasks fully.

Task 1 and Task 2.

To achieve a **HIGHER PASS (41 – 69%)** you need to complete the following tasks fully.

Tasks 1, 2, 3 and 4

To achieve a **FIRST CLASS MARK (70 – 85%)** you need to complete the following tasks fully.

Task 5

To achieve **THE HIGHEST LEVEL OF THE MARKS (86 – 100%)** you need to complete the following tasks fully.

Task 6 and Task 7.

**Submission**

Via Blackboard by 2pm on the 22nd of March 2018

One zipped folder labelled "DADSA Coursework B <your name & student number>"

All data used to run the system must be included within the submitted folder

Note that it is your responsibility to ensure that you have submitted a folder that contains all the required material and one that can be opened and read by software that are available at UWE