You will design the behaviour of complex programs involving the fundamental programming constructs – functions, operations,

The prestigious football club who originally hired our Software Development company is very pleased with the outcome and has requested an extension. The program already been crafted with you which I have shared to shezam with a title “Simple-code” takes in rates of 6 skills of new players in the form of a score between 0 and 5 (shouldn’t take inputs outside of this range). The six main criteria to rate the skills of the player should be:

1. Speed  
2. Shooting

3. Passing  
4. Defending

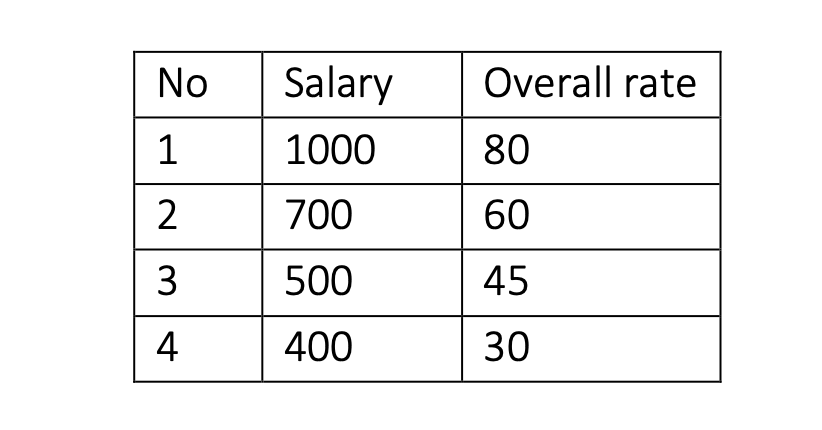
5. Dribbling

6. Physicality

The program uses these individual skill rates to calculate an overall rating out of 100 using the formula:

*overall rate = summation of skills rate \* 100 / 30*

he resulting number (should be between 0 and 100) is then used to estimate the salary using the salary dataset:

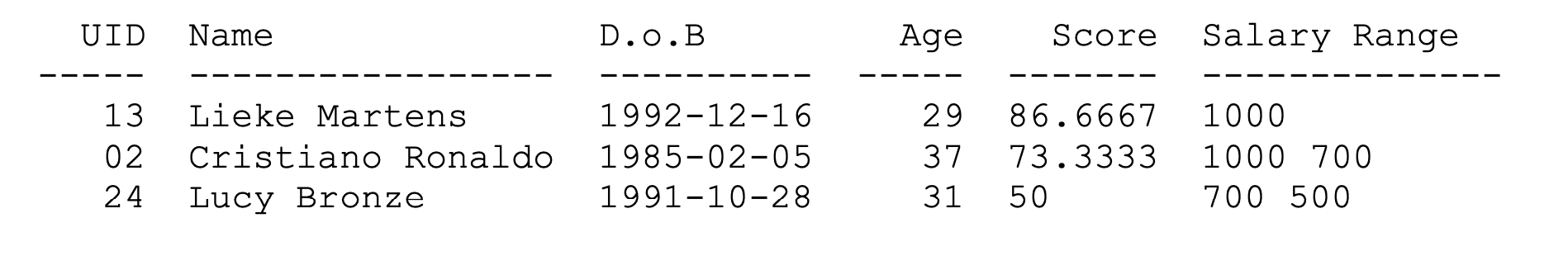


From the dataset above,

* If the overall rate is greater than or equal 80, the salary should be: 1000
* If the overall rate less than or equal to 30, the salary should be: 400
* If the overall rate is in between two numbers, like 45 and 30, the output should be a range of the maximum salary and the minimum salary: 500 400

You are asked to add the following functionality:

1. Functions: Split and pack up our given code into functions. The program must contain a main() function and at least two other functions called “calculate\_rating()” and “calculate\_salary()”.  
   Use the main() function to call the other functions to receive the 6 player’s skill ratings, calculate the player’s score and their salary range.
2. Input expansion: Adjust your program to ask users to enter a player ID (2-digit number), name and date-of-birth (D.o.B) by sequence, before asking for the 6 skills from our given code (speed, shooting, passing, defending, dribbling, physicality). D.o.B should be in ISO format YEAR-MONTH-DATE.
3. Loop your code: Your program should use a loop to ask for the information of three players, or until the input for user ID is “end”.
4. Calculate the players' overall ratings and store it for later use.
5. Calculate the players' salary ranges and store it for later use.
6. Calculate players' ages: According to each user’s year of birth, your program should calculate the age of the player and store it for later use. Hint: the library datetime can be used here to make your job easier.
7. Tabulate: Display a table that summarises the input data (ascending order with user ID) using the tabulate library. It should look like:



1. File writing: Save the table into a new local file named “players.txt”
2. Make sure that you:
   1. Implement robust input validation for a player's ID, D.o.B., and ratings. Your program should give a warning that says *"The rating you entered was invalid"* for any input errors.
   2. Display use of appropriate data structures for storing player information.
   3. Display use of appropriate code structures for implementing the functionality.
3. Advanced Function: Once everything in requirements 1-9 has been completed, extend your program with one more function called advanced(). This function should:
   * 1. be a complimentary main() method that, instead of taking user input, uses the provided file as input.
     2. accept in its argument (filename) the name of the text file to read.
     3. read the user information record from the provided file “PlayerData.txt”, which contains player IDs, names, and date-of-births. The function should then use this data as the input for the program.
     4. call all the other functions used and produces a text file output (requirement 8).