TU Dublin - Tallaght Campus
Department of Computing
Software Development 2
Repeat Assessment (60%)
Please note rules regarding plagiarism

Instructions:
Include your name and student number as a comment at the start of your program.

Upload a zipped folder to Moodle containing the .py file , .csv files and Test Tables.

Attempt all parts.
It is your responsibility to ensure that you upload the correct version of your program.
Marks will be awarded for successful compilation and a working program according to the specification, good program design, style and comments.

Problem:

Write a Python program for a college student system.

Create a **students.csv** file in Excel and save it to your PyCharm project folder in order to use it in your system. This is a **csv** file where each line represents a student's details: in each line/row:
the first cell/value is the student ID, in the format of X123456, i.e. X followed by 6 digits
the second cell/value is the student name, e.g. John Smith
the third cell/value is the student 's date of birth, in the format dd/mm/yyyy
the fourth cell/value is the student's password –for students entered via Excel give this a value of "Empty"
the fifth cell/value is the number of tests that the student has taken, which can only be either 1,2 or 3
the sixth cell/value is the average result that the student has scored over all the tests they took, values in the range of 0..100, (put any value in here within the permissible range)

Sample Rows:

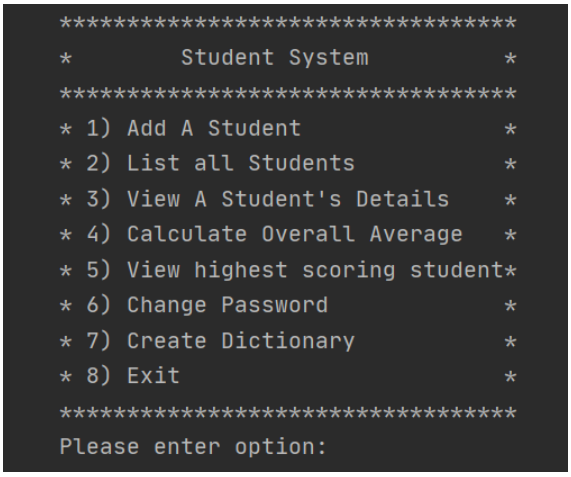
X123456	John Smith	02/11/1998	Empty	3	65
X234567	Mary Smith	01/02/1999	Empty	2	45

For your system:
Create a **static** class **StudentOptions** :

with a **class** variable, VALID_LENGTH with a value of 7, which represents the valid length of a password in the system.

Code the following **static** methods in the **StudentOptions** class:

A **menu()** method that displays a menu as follows:



An **add_student()** static method, it takes the 2D list of students as a parameter, then reads in the following from the user:

- the ID number of the student:
  This ID should be passed to a **validate_id()** method, which you need to write, which returns True or False depending on whether the ID is a valid ID or not. A valid ID is an ID that is of length 7 and contains no spaces.
  Your code should loop until a valid ID is entered for the student.
- the name of the student,
- the date of birth of the student,
  The date and month of the date of birth should be validated, create appropriate methods, so that the date is appropriate for the month entered, e.g. there are only 30 days in September so 31/09/1999 is invalid. Loop until a valid date is entered. You do not need to validate the year.
- the number of tests taken, validate so that only 1 or 2 or 3 is entered
- the average result of the student, validate so that a value of between 0 .. 100 inclusively is entered
- A password should be generated for the student in a **generate_password()** method, (return it from the method), by passing the student ID to the **generate_password()** method and based on the following criteria:
  - The first character of the new password should be the last digit in the ID
  - Then looping through the digits in the ID from left to right up to but not including the last digit, multiply each digit by the value of the original last digit and calculate the modulus of that value, that result becomes the next digit in the new password.
  - Finally add the first character of the original ID, i.e. “X” as the last character in the new password.
  - For example: a student with an ID of “X123456” generates a password of “662840X”

When all the data has been entered and the password generated for the student, save that student’s details to the 2D student list.

A **list_students() static** method that takes the 2D list of students as a parameter and displays the students' details in a formatted manner.

If there are no students in the 2D list your method should print an appropriate message, otherwise display the students' details appropriately.

A **view_student_details() static** method that takes the 2D list of students, and a student ID as parameters and displays the details of that student in a formatted manner. If the student ID is not found in the list an appropriate message should be displayed to the user.

A **calculate_overall_average() static** method that takes as a parameter the 2D list of students, and calculates the overall average of all the students. A count should then also be made of the number of students whose average is higher than the overall average and this count printed to the screen.

A **highest_student() static** method, that takes as a parameter the 2D list of students, determines and prints the details of the student who scored the highest average. If there are multiple students who scored the same highest average, the details of each of these students should be printed to the screen

A **change_password() static** method, which takes as parameters, the ID of the student for whom the password is changing and the 2D list of students, and allows the user change the existing password. In order to change the password, you should first find the student in the list, if the student's ID is not in the list an appropriate message should be printed, otherwise the user must enter the existing password and this must match with the stored password, only then is the user allowed to enter a new password and change the existing password. The newly entered password should be validated as before, i.e. only 7 characters and no spaces.
The student's password should then be updated.

The most frequent request from users of the system is to generate a list of student IDs and their corresponding average. In order to generate faster access to these values, write a **static create_dictionary**() **method**, which takes as a parameter the 2D list of students: in the method create a dictionary with the student ids as the keys and their corresponding average as their values.
Then, in the method, display the dictionary key-value pairs of the student ID and their average result:

In the main body of the program, implement the following logic:

The csv file should be read into the main body of your code as a 2D list.

The program should then implement a menu driven system with the menu being displayed and the user entering the options required until the Exit option, 8, is selected.

Invalid options should be caught.

If **option 1** is selected:

The add_student() method as outlined above to add a student should be called, with the 2D list of students passed in as an argument.

If **option 2** is selected:

The list_students () method should be called with the 2D list of students passed in as an argument.

If **option 3** is selected:

The ID of the student to be viewed should be read in from the user and passed with the 2D list of students as parameters to the view_student_details() method.

If **option 4** is selected:

The calculate_overall_average() method should be called.

If **option 5** is selected:

The highest_student() static method should be called.

If **option 6** is selected:

The ID of the student for whom the password is to be changed is entered by the user and passed with the 2D list of students to the change_password() static method.

If **option 7** is selected:

Call the create_dictionary() method, passing the 2D student list to it.

After the user has selected the exit option to exit the menu system, the program should write the 2D list of students to a csv file called **newstudents.csv**.

Test your program by creating a set of appropriate test tables for it.