Foundations of Programming 2018-19

Term-2 Assignment – 100 points (15% of the module)

Implement a simple Database Management System (DBM) in Python (version 3.5), which enables the user to create and maintain a small database of student records.

PART A (45 marks) – Requirements specification and Interface

Your program will contain a function `main()`, which should visualize a set of 4 possible options (see below) and prompt the user to make a choice. Depending on the choice entered, the function should perform a different action, as follows:

Choice **PROGRAM'S BEHAVIOUR**

A (**Add student**): when this option is chosen, the program should ask the user to enter a new student's name and surname (in that order) as two separate strings. Then, a new student record should be created and added to the current list of students (the "database"). If the database *already contains a student with the given name and surname*, the program should ***not crash***, but print a warning message (e.g., "Student <…> already exists") and return to the main menu, ***leaving the database unchanged.***

[10 marks]

R (**Remove student**): when this option is chosen, the program should ask the user to enter name and surname of an existing student; if found in the database, the corresponding record should be ***removed*** from it. If the database is **empty** or does not contain the specified student, the program should **not crash** but print a warning message (e.g., "Student not found", or "Database empty") and *return directly to the main menu*, leaving the database unchanged.

[10 marks]

L (**List database**): the program should print the full list of students currently in the database, with each student's *surname* and *name* printed (in this order) on a separate line. If the database is empty, the program should **not crash** but print a warning message (e.g., "Database empty"). **[5 marks]**

X (**EXIT)**: the program should terminate gracefully, i.e., without crashing or producing an error.

[5 marks]

After executing the appropriate action for the chosen option, the program should prompt the user for a new choice. ***This behaviour should be repeated until the user chooses option "X".***

NOTE: ***your interface should adhere as closely as possible to the above specifications. Any differences (e.g., using "a" instead of "A") will lead to penalties of up to 40 MARKS.***

ERROR CHECKING

If the user enters something which is not one of the allowed options (A, R, L, X), your software should NOT crash. Instead, it should ignore the invalid input and prompt the user for a new choice. Similarly, the system should be able to respond adequately (i.e., ***not crash***) in cases when the user enters an empty string (or a number) instead of a name / surname.

[15 marks]

IMPORTANT: THE USE OF ANY PYTHON MODULES IS FORBIDDEN

The presence of an "*import*" statement in your code will carry a penalty of up to 80 MARKS.

P.T.O.

PART B (30 marks) – Requirements specification

At Hypothetical University (who commissioned your software), marks are on a 4-point scale (**A** = 4.0 points, **B** = 3.0, **C**=2.0 and **D**=1.0) and each course is valued in terms of **credit hours**. Each student's final grade is computed using *"Quality Points"* (**QP**s). E.g., if a course is worth 30 credit hours and the student gets an A (= 4.0 points) on it, (s)he earns 30 * 4.0 = 120 QPs.

Extend the code written for Part A so that the menu includes the following additional choice:

Choice PROGRAM'S BEHAVIOUR

G (**Add Grade**): when this option is chosen, the program should first ask the user to enter an existing student's name and surname as two separate strings. If the student is in the database, the record should be updated, as follows: the program should **first ask the user to enter a grade** (i.e., one of 'A', 'B', 'C' or 'D'). **Then, it should ask the user to enter a float** (the credit hours for that course). The system should then add the resulting QPs to the student's record. If the database is empty, or does not contain a student with the name / surname the user provided, the code should ***not crash*** but print an appropriate warning message and *return directly to the main menu, without asking the user to enter grade or credit hours*.

[20 marks]

ERROR CHECKING and INTERFACE

Error checking and interface should be analogous to those in Part A (except that now option 'G' is allowed). I.e., in such cases as: an empty string, or a number, are entered as a student's name or surname; a string, or a negative number, or zero, are entered for credit hours, the program should ***not crash***, but print a short warning message, and prompt the user again. **Similarly,** when the user is asked to enter a student's new grade, if the grade entered is invalid (i.e., it is not one of 'A',... 'D', or it is the empty string, or a number, etc.), the program should not crash but print a warning message (e.g., "Invalid grade") and **ask the user to enter the grade again.** *This behaviour should be repeated until a correct grade is entered*, only after which the program is allowed to move on and ask the user to enter the credit hours.

[10 marks]

***To avoid penalty marks, ensure the interface adheres closely to the above specifications*.**

PART C (25 marks) – *This part requires PART B to have been completed*

Modify the code written for Part B so that option "L" produces the following behaviour:

Choice PROGRAM'S BEHAVIOUR

L (**List**): print all students in the current database. For each, print surname, name, and Grade Pont Average, GPA (see below), *in this order.* Each student to be printed on a separate line.

[15 marks]

The list of students should be printed in *descending* GPA's order (i.e., best students first).

[10 marks]

A student's GPA is their **total QPs** divided by the **total credit hours** completed. For example, if a student has accumulated 155 QPs by taking several courses worth 55.0 credit hours in total, his GPA should be 155 / 55.0 = 2.81818181... (or, rounded up to 2 decimals: 2.82).

NB: if the database is *empty*, your code should **not crash** but print a warning message and return to the main menu. If there are students with ***no grades***, choosing 'L' should ***not*** cause the code to crash, but the value "None" to be printed instead of the GPA for those students.

IMPORTANT: THE USE OF ANY PYTHON MODULES IS FORBIDDEN

The presence of an "*import*" statement in your code will carry a penalty of up to 80 MARKS.

TESTING YOUR CODE

Parts B relies on having correctly completed Part A, and Part C relies on Part B. Before moving on to Part B (or C), you should test the correctness of your Part A's (or B's) solution. Below are examples of interactive sessions which your system should be able to replicate as shown:

EXAMPLE TESTING for PART A:

Explanation	***Program's on-screen Output / User Input***
<The user is prompted to make a choice >	`Choose A, R, or L ('X' for exit):`
<The user enters 'A' and hits "Enter">	`A`
<The user is prompted to enter a name>	`New student's name:`
<The user enters "John" and hits "Enter" >	`John`
<The user is prompted to enter a surname >	`New student's surname:`
<The user enters "White" + Enter>	`White`
<The system outputs a confirmation message>	`Student John White has been added`
	`Choose A, R, or L ('X' for exit):`
<The user chooses 'A' >	`A`
<The user is prompted to enter a name >	`New student's name:`
<The user enters "Mary" + Enter >	`Mary`
<The user is prompted to enter a surname >	`New student's surname:`
<The user enters "Smith" + Enter>	`Smith`
<The system outputs a confirmation message>	`Student Mary Smith has been added`
	`Choose A, R, or L ('X' for exit):`
<The user chooses 'L'>	`L`
<The program prints the current database>	`White, John`
	`Smith, Mary`
	`Choose A, R, or L ('X' for exit):`
<The user chooses 'R'>	`R`
<The user is prompted to enter a name>	`Name of the student to be removed:`
<The user enters "John">	`John`
<The user is prompted to enter a surname>	`Surname of student to be removed:`
*<The user **erroneously** enters "Smith">*	`Smith`
<The system issues a warning message… >	`No John Smith in database!`
*<… and returns **directly** to the main menu>*	`Choose A, R, or L ('X' for exit):`
<The user chooses 'R' again>	`R`
<The user is prompted to enter a name>	`Name of the student to be removed:`
<The user enters "John">	`John`
<The user is prompted to enter a surname>	`Surname of student to be removed:`
<The user enters "White">	`White`
<The system outputs a confirmation message>	`Student John White has been removed`
	`Choose A, R, or L ('X' for exit):`
<The user chooses 'L'>	`L`
<The program prints the current database >	`Smith, Mary`
	`Choose A, R, or L ('X' for exit):`
<The user chooses 'X'>	`X`
<The program gracefully terminates>	`Thank you – Goodbye.`

TESTING YOUR CODE

EXAMPLE TESTING for PART B:

Explanation	***Program's on-screen Output / User Input***
<Assume the database contains students John White and Mary Smith, as from above Part-A testing>	
<The user is prompted to make a choice >	Choose A, R, L or G ('X' for exit):
<The user chooses 'G' >	G
<The user is prompted to enter a name>	Student's name:
<The user enters "John" + Enter >	John
<The user is prompted to enter a surname >	Student's surname:
<The user enters "White">	White
<The user is prompted to enter a new grade>	Please enter a grade (A/B/C/D):
*<The user **erroneously** enters '4.0' >*	4.0
<The system issues a warning message…>	Invalid grade!
<… and asks the user again to enter a grade>	Please enter a grade (A/B/C/D):
<The user enters "A" >	A
<User is prompted to enter the credit hours>	Please enter credit hours (>0):
<The user enters "25.0">	25.0
<The system outputs a confirmation message>	100.0 QPs added to White, John
	Choose A, R, L or G ('X' for exit):
<The user chooses 'G' >	G
<The user is prompted to enter a name>	Student's name:
	John
<The user is prompted to enter a surname >	Student's surname:
	White
<The user is prompted to enter a new grade>	Please enter a grade (A/B/C/D):
<The user enters "C" >	C
<User is prompted to enter the credit hours>	Please enter credit hours (>0):
<The user enters "35.0">	35.0
<The system outputs a confirmation message>	70.0 QPs added to White, John
	Choose A, R, L or G ('X' for exit):
<The user enters 'L'>	L
<The program prints the database, with GPAs>	White, John – GPA: 2.83
	Smith, Mary – GPA: None
	Choose A, R, L or G ('X' for exit):
	X
	Thank you – Goodbye.

***NOTE**: A different set of tests will be used to assess your code. If your program produces the expected outputs in the above examples, it does not necessarily mean that it is correct, or that it will score full marks.*