University of Glasgow | School of Computing Science

Individual Assessment

Course Name	**CS1P**			
Coursework	**Individual Assessment (replacing lab exam)**			
Deadline	**Time:**	**16:00**	**Date:**	**25 March 2020**
% Contribution to final course mark	**10**			
Solo or Group ✓	**Solo**	✓	**Group**	
Anticipated Hours				
Submission Instructions	Moodle page for CS1P: https://moodle.gla.ac.uk/mod/assign/view.php?id=1467618			
Please Note: This Coursework cannot be Re-Assessed				

Code of Assessment Rules for Coursework Submission

Deadlines for the submission of coursework which is to be formally assessed will be published in course documentation, and work which is submitted later than the deadline will be subject to penalty as set out below.

The primary grade and secondary band awarded for coursework which is submitted after the published deadline will be calculated as follows:

(i) in respect of work submitted not more than five working days after the deadline
   a. the work will be assessed in the usual way;
   b. the primary grade and secondary band so determined will then be reduced by two secondary bands for each working day (or part of a working day) the work was submitted late.

(ii) work submitted more than five working days after the deadline will be awarded Grade H.

Penalties for late submission of coursework will not be imposed if good cause is established for the late submission. You should submit documents supporting good cause via MyCampus.

Penalty for non-adherence to Submission Instructions is 2 bands

You must complete an "Own Work" form via https://studentltc.dcs.gla.ac.uk/ for all coursework

CS1P - INDIVIDUAL ASSESSMENT (replacing lab exam 2)

Individual Assessment Overview

The individual assignment for CS1P is a take-home replacement for Lab Exam 2. This assessed exercise is worth 10% of the final course grade. Please follow the instructions *very carefully* to ensure your submission is delivered to us safely.

The Problem

Personalised recommendation services are part of many websites. Amazon, YouTube, and Facebook are all examples of sites that suggest potential items (e.g., products, videos, etc.) of interest to users based on their previous activity on the site. This can be handy for users, but also very important for businesses.

The task of this assessment is to implement a program that makes personalised book recommendations using a similarity algorithm to predict which books a user is likely to enjoy reading based on their past or sample ratings.

1. Input Files

Your program should read data from two input files:

`books.txt` includes a list of 55 books in an `<author>,<title>` format, one entry per line

`ratings.txt` includes usernames to represent users of the service followed by a list of 55 integers, each representing a rating for each book from `books.txt`, in the same order. The file is structured using the following format: `<user_a>\n<user_a_rating_1> … <user_a_rating_55> \n`

2. Ratings

Each integer rating should be interpreted as follows:

Rating	Meaning
-5	Hated it!
-3	Didn't like it
0	Haven't read it
1	OK
3	Liked it!
5	Really liked it!

3. The similarity algorithm

If `User A` enjoyed reading a number of books that `User B` also enjoyed reading, then when `User A` recommends another book that `User B` has not read, chances are that `User B` would like to read it too. On the other hand, if the two users always tend to disagree about books, then `User B` is unlikely to read a book that the `User A` recommends.

A program can calculate how similar two users are by treating each of their ratings as a vector and calculating the dot product of these two vectors. The dot product is simply the sum of the products of each of the corresponding elements.

For example, suppose we had 3 books in our database and `User A` rated them [5, 3, -5], `User B` rated them [1, 5, -3], `User C` rated them [5, -3, 5], and `User D` rated them [1, 3, 0]. The similarity between `Users A` and `B` is calculated as: (5 x 1) + (3 x 5) + (-5 x -3) = 5 + 15 + 15 = 35. The similarity between `Users A` and `C` is: (5 x 5) + (3 x -3) + (-5 x 5) = 25 - 9 - 25 = -9. The similarity between `Users A` and `D` is (5 x 1) + (3 x 3) + (-5 x 0) = 5 + 9 + 0 = 14. We see that if both people like a book (rating it with a positive number) it increases their similarity and if both people dislike a book (both giving it a negative number) it also increases their similarity.

Once you have calculated the pair-wise similarity between `User A` and every other user, you can then identify whose ratings are most similar to `User A`'s. In this case, `User B` is most similar to `User A`, so we would recommend to `User A` the top books from `User B`'s list that `User A` hasn't already read.

4. The complete program

Write a program that takes users' book ratings and makes recommendations to them using the similarity algorithm above. If the user for whom we want to provide recommendations has already rated books in the database (i.e., the username already exists in `ratings.txt`), then similarity with other users should be calculated based on her/his existing ratings. Otherwise, the program should ask the new user to rate about 20% of the titles selected at random (from `books.txt`), store the new user's username and ratings in `ratings.txt`, and then calculate similarity and make recommendations based on the user's input.

The program should return a number of recommendations (specified by the user) of books that the user has not already read, and other user(s) with a high similarity score have rated positively (i.e., with 3 or 5). If the required number of recommendations cannot be compiled from the single user with the highest similarity score (because, e.g., both users have read the same books already), then the list should be completed with recommendations from user(s) with the next highest similarity score, and so on.

5. User Input and Output

You are NOT required to provide a Graphical User Interface for this exam. A Command Line Interface is sufficient. If you want, however, the extra challenge of providing a GUI environment you are allowed to do so.

The program should ask for a username and the number of recommendations required. The default should be 10. If the username does not already exist in the database, then a random sample of ratings should be requested from the user as specified above. The list of recommendations should include titles rated positively by other user(s) with high similarity score(s).

The program should print the list of recommended titles, also indicating which titles have been recommended by which user. Finally, this recommendation list should also be written to `output.txt`. An example can be found in `sample_output.txt`

6. Testing

The `Assessment_setup_files` folder on moodle will include the two input files (`books.txt; ratings.txt`) and a single placeholder (`recommendations.py`) file for you to write your code in. A placeholder output (`output.txt`) file will be included for you to write an indicative list of 10 recommendations for a user, as a sample output of your program.

You should perform extensive error checking for your program (such as, e.g., incorrect input, non-existing files, invalid ratings, etc.). You are free to use either defensive programming or exception handling.

Suggestions on how to progress

1. Choose appropriate data structure(s) to use and code/function(s) to process the input files
2. Think carefully about the design of your program. Use the necessary functions for your code to be readable, your overall implementation efficient, and your error checking adequate.
3. Think carefully how many data structures you will use and how to manipulate them in order to produce the desired result.
4. If you choose to implement the GUI, carefully plan the interaction between `gui.py` and `recommendations.py` in terms of function calls and parameter passing

What you must do and what you must submit

You may use any standard Python modules, functions and methods. For example, the `split` method of a string and functions from the `random` module are likely to be useful.

The following files will be available in `Assessment_setup_files` which can be downloaded from Moodle:

- `recommendations.py`: Placeholder for the main program

- `books.txt`: Input file with the list of book titles
- `ratings.txt`: Input file with the list of user ratings
- `output.txt`: Placeholder for your sample output.
- `sample_output.txt`: Sample the content that your program should write to `output.txt`.

You **must** submit these files:

1) `recommendations.py`: which contains your program. Submitting in any other format will result in a **zero**.
2) `output.txt`: which contains the output by your program in a format similar to `sample_output.txt` (`sample_output.txt` can be downloaded from moodle). **IMPORTANT**: this file **must** include the following information after submission:
    - The name of the user for whom these recommendations are made. For testing purposes, you should choose **one of the existing users** from `ratings.txt`
    - The list of 10 recommendations **CLEARLY** indicating **which books** are recommended by **which other user**.
3) `explanation.txt`: which contains an explanation of how you solved the problem. The template can be downloaded from moodle. You only need to answer the questions in there.
4) `gui.py` (optional): if you choose to implement a GUI, then you are allowed to submit an additional file.

Marking guidelines

The following aspects of the program will be taken into account in marking:

- use of appropriate and correct algorithms and Python control structures.
- program efficiency.
- use of appropriate data structures.
- correct Python syntax.
- good programming style, including layout, use of explanatory comments, choice of identifiers, and appropriate use of functions.
- error checking; you are free to use exception handling or defensive programming.
- correctness of the implementation on unseen test input.

Plagiarism

Plagiarism is taken very seriously in this assessment. We will run similarity check on all submissions. If similarities arise, they will be checked by the course lecturer. Confirmed cases of plagiarism will be reported to the Senate. Penalties depend on the severity of the case, and will range from receiving a zero for the assessment to expulsion.

More information below:

- What is Plagiarism and what happens if you plagiarize: https://www.gla.ac.uk/myglasgow/leads/students/plagiarism/

- University's Plagiarism statement: https://www.gla.ac.uk/myglasgow/senateoffice/policies/uniregs/regulations2019-20/feesandgeneral/studentsupportandconductmatters/reg32/

Feedback

After publishing the grades, you will be able to collect your marked submission and the marker's feedback from the student support and services office in the ground floor of Sir Alwyn Williams Building.

Marking Scheme

	Solution is working and well explained.	Interactive input and error checking.	Correctness, clarity, and efficiency of the code
Weight	10/20	5/20	5/20