Assessment Proforma 2024-25

Key Information

Module Code	CM1208
Module Title	Maths for Computer Science
Module Leader	Prof. Yukun Lai
Module Moderator	Prof. Stuart Allen
Assessment Title	Implementation of a simple e-commerce recommendation program
Assessment Number	1 of 2
Assessment Weighting	50% of a 10-credit level 4 module
Assessment Limits	N/A

The Assessment Calendar can be found under ‘Assessment & Feedback’ in the COMSC-ORG-SCHOOL organisation on Learning Central. This is the single point of truth for (a) the hand out date and time, (b) the hand in date and time, and (c) the feedback return date for all assessments.

Learning Outcomes

The learning outcomes for this assessment are as follows:

1. Show awareness of different aspects of mathematics in analysing and understanding important areas of computing
2. Appreciate how mathematical techniques contribute to the study of computing
3. Apply mathematical techniques and knowledge to a problem or situation
4. Demonstrate an awareness of solving practical problems using mathematics and Python programming

.

Submission Instructions

The coversheet can be found under ‘Assessment & Feedback’ in the COMSC-ORG-SCHOOL organisation on Learning Central.

All files should be submitted via Learning Central. The submission page can be found under ‘Assessment & Feedback’ in the CM1208 module on Learning Central. Your submission should consist of multiple files:

Description		**Type**	**Name**
Coversheet	Compulsory	One PDF (.pdf) file	[student number]_Coversheet.pdf
Python code	Compulsory	Main Python source code	Recommend.py
Python code	Optional	Additional Python source files	No restriction

Any code submitted will be run in Python 3 (Windows) and must be submitted as stipulated in the instructions above. Any deviation from the submission instructions above (including the number and types of files submitted) may result in a reduction in marks for the assessment.

If you are unable to submit your work due to technical difficulties, please submit your work via e-mail to comsc-submissions@cardiff.ac.uk and notify the module leader.

Assessment Description

Write a Python application program (main file called **Recommend.py**) that implements e-commerce recommendation based on item-to-item collaborative filtering, as discussed in the lectures.

Input and Output. The input to your program involves two text files **history.txt** which contains the purchase history of all the customers and **queries.txt** which includes all the queries (shopping cart items based on which you need to provide recommendation).

The file **history.txt** is a text file describing the complete purchase history of all the customers. The first line includes three numbers:

Number_of_Customers Number_of_Items Number_of_Transactions

This is followed by *Number_of_Transactions* lines of text, each line containing two numbers:

Customer_ID Item_ID

This means the customer *Customer_ID* bought the item *Item_ID*. Both *Customer_ID* and *Item_ID* are integers starting from 1. Note that the same customer may have bought the same item multiple times.

The file **queries.txt** is a text file with each line containing a query, describing a list of items in the current shopping cart. Each query is composed of one or more numbers, separated by whitespace, corresponding to the item IDs in the current shopping cart. All the input files described above are assumed to be in the current folder. Your program should read from these files *in the current folder* without any user interaction. Do *not* ask users to enter paths or filenames.

For output, print the required messages (see details below and sample outputs overleaf) following exactly the same format (except for the actual number of whitespace) as specified and as in the examples. When printing real numbers, you should print them with at least two fractional digits.

Reading the transaction history. Your program should read in all the transactions from **history.txt**, and build the customer-item purchase history table (as explained in the lecture) where an entry of 1 means the customer has bought the item and 0 otherwise. Print the total number of non-zero entries (i.e. with a value of 1) in the customer-item purchase history table ("Positive entries: *number*").

Precomputing item-to-item angles. Your program should work out the angles (in degrees) between every pair of items (excluding an item with itself). Print the average of all the pairwise angles ("Average angle: *average angle*").

Recommendation. For each query (each line in **queries.txt**), your program should perform the following:

- Print the query in a line "Shopping cart: *query*".
- For each item *Item_ID* in the query (in the order as it appears), find an item *Match_ID not in the current shopping cart* which has the minimum angle *Min_Angle* with the item *Item_ID*.
  - If *Min_Angle* is less than 90°, print "Item: *Item_ID*; match: *Match_ID*; angle: *Min_Angle*".
  - Otherwise, no match is accepted, and you should simply print "Item: *Item_ID* no match".

  Note that when multiple items have the same minimum angle, your algorithm can print any one of these. Each printed item *Match_ID* is considered as a candidate for recommendation.
- Produce the recommendation list by combining all the candidates and order them in increasing order of angles. For items which are considered relevant via different shopping cart items, the minimum angle it makes with any item in the shopping cart should be used in ranking, and it should only appear in the recommendation list once. If multiple candidates have the same angle, they may appear in arbitrary order. Print "Recommend: *list of recommended items*".

Error checking is *not* required (invalid input, etc.) *Remember that **I will perform extensive tests on your program, using more than just the test data I have provided**.*

Hint: The Week 5 exercise sheet includes exercises related to handling text files using Python, which can be useful. Several further test cases and their expected output are provided on learning central.

Assessment Criteria

Assessment and marking of your program will be done by automatically checking the output of your program against my program on a large set of test conditions. This will be done using a text comparison tool, so make sure your outputs match mine as formatting of text will be critical!

Note that extra blank spaces are ignored (so are acceptable), and when printing numbers you may print more fractional digits than the example shown below.

An implementation that does not exactly follow the assignment instructions, including text formatting not matching the description and examples shown in the assignment, may result in a deduction of up to 50% of marks for the relevant features.

The breakdown of marks will be:

- 10% – reading in purchase history
- 25% – precomputing item-to-item angles
- 5% – reading and printing queries
- 30% – working out candidate for each item in the shopping cart
- 15% – printing recommendation in the correct order
- 15% – efficiency (i.e. excessive run-times will be penalised)

High 1st 80%+	Code fully working or with very minor errors. Code is reasonably efficient.
1st 70-79%	Code contains some minor errors. Code may be less efficient.
2.i 60-69%	Code is mostly working, but contains clear mistakes for certain steps. Code may be less efficient.
2.ii 50-59%	Code is mostly working, but contains major mistakes for multiple steps. Code may be less efficient.
3rd 40-49%	Code is mostly complete but contains major errors or only some features are attempted in the code. Code may be less efficient.
Marginal Fail 30-39%	Code is incomplete or contains substantial errors.
Fail 0-29%	Code does not work or only a small number of features are attempted.

Help and Support

Questions about the assessment can be asked on https://stackoverflow.com/c/comsc/ and tagged with ‘CM1208’, or at the beginning of the lectures in *Weeks 6, 7, 8, 9*.

Support is also available via email/MS Teams meetings upon request.

Feedback

Feedback on your coursework will address the above criteria. Feedback and marks will be returned on Learning Central, supplemented with cohort feedback.

Feedback from this assignment will be useful for follow-up modules where Python programming is involved.