CAB203
Problem Solving Tasks

1 Introduction

In these problem solving tasks you will demonstrate your knowledge and skill with the material in the unit by developing solutions to real world problems by translating them to mathematical language, relating the problems to well known problems on mathematical structures, and implementing software to solve the problems.

Your submission will consist of two parts: a report detailing the mathematical descriptions of the problems and solutions, and a Python file containing your implementation of the solutions.

This assessment item includes an option to use generative AI, such as ChatGPT. Please see section 5 for more information.

This assignment is worth 30% of your final grade.

Please see the Canvas Assignments page for due dates.

2 Tasks

Choose ***one*** out of the following three tasks to solve.

2.1 Regular languages and finite state automata

This task will simulate a simple client-server protocol, analogous to what might happen with a connection to a service like FTP. Your job is to write a small server program that receives messages from a client and performs some simple computations. You won't need to implement the network stack or anything, just the behaviour of the server based on the commands that it receives. The behaviour is as follows:

- If the server is not connected to a client and receives a message `connect` then it responds with `ready` and is now connected to the client
- If the server is connected to a client and receives a message `connect` then it responds with `busy` and remains connected to the client (you can imagine that this comes from a second client, but we will not have any way of telling where the messages come from.)
- If the server is connected to a client, is not in one of the special modes below, and receives a message `disconnect` then it responds with `goodbye` and is no longer connected to the client

- If the server is connected to a client and receives a message `mailbox` then it responds with `enter email address` and enters mailbox mode.
- If the server is in mailbox mode and receives a valid email address (according to the limited criteria from Tutorial 10, Section 2, Question 5b) then it responds with the mailbox part of the email address. The server remains connected to the client but exits mailbox mode.
- If the server is connected to a client and receives a message `domain` then it responds with `enter email address` and enters domain mode.
- If the server is in domain mode and receives a valid email address (according to the limited criteria from Tutorial 10, Section 2, Question 5b) then it responds the domain part of the email address. The server remains connected to the client but exits domain mode.
- In all other circumstances the server responds with `error`. Connection status and mailbox/domain modes are not affected.

Your server program will be implemented as a single Python function:

```
def emailParserBot(message, state):
    # your code here
    return response, state
```

Note that your function takes two arguments: the incoming message and a state. `state` is for you to keep track of connection status and modes. You are responsible for initialising `state` and changing its value as appropriate. You can use whatever type you like for `state`. On the very first call to your program `state` will be set to `None` to indicate the start of a new testing session (as though the server were rebooted).

Your function should return two values. The first return value (`response` in the above) must be a string according to the behaviour listed above. The second return value is the new value of `state`, calculated as you see fit. This value will be provided as the `state` argument in the next call to your function.

Please note: your program should not use any global variables. All state information should be kept in the `state` argument/return value. Using global variables may cause problems with the testing system.

Your program must use the `re` library. Your report must include a state change diagram depicting the state changes in the behaviour described above.

2.2 Linear algebra

Dana needs to select fertiliser for her crops. She has access to several different types of fertiliser, each containing a different mix of nitrogen, phosphorus, and potassium. Meanwhile, each crop has particular needs for each of these minerals. Dana has access to test equipment and tables from which she can determine the total amount of nitrogen, phosphorus and potassium required for each of her fields. As well, she has recorded the percentage of each mineral present in each type of fertiliser in a CSV file organised like so:

```
Fertiliser,Nitrogen,Phosphorus,Potassium
```

```
All purpose,7,1,6.5
Chicken manure,1.2,1.2,1.8
Cattle manure,0.5,8.5,0.95
```

There will always be exactly three types of fertiliser in the CSV file.

Your task is to determine the amount of each type of fertiliser that Dana should use to achieve the required amount of each mineral, or decide that it is not possible to do so.

Your implementation should be a function `fertiliser(csvfilename, n, p, k)` where:

- `n` is the amount of nitrogen required by the crop in kg
- `p` is the amount of phosphorus required by the crop in kg
- `k` is the amount of potassium required by the crop in kg

Your function should return a dictionary containing the required amounts of each fertiliser in kg, like so:

```
{
   'All purpose': 25.2,
   'Chicken manure': 17.5,
   'Cattle manure': 22.8
}
```

If there is no solution to the problem your function should instead return `None`. Dana's scale measures to the nearest 0.01kg. Therefore your answers should be rounded to 2 decimal places. You can use the `round` function for this. Since floating point numbers, used by `numpy`, are not infinitely precise, the test cases will tolerate up to 0.01 difference in the fertiliser amounts.

Dana cannot add a negative amount of fertiliser to her field, so if one of the fertiliser amounts is negative (after rounding) then that does not count as a solution to the problem. Likewise, all the arguments to the function will make sense physically, i.e. they will all be non-negative. Mathematically, it is possible for there to be an infinite number of solutions, but this is beyond what we studied in the unit, so such cases can be ignored; they will not be used as test cases.

Your Python function must use `numpy`.

Hint: This problem can be modelled as a set of three linear equations in three unknowns. We didn't study this case explicitly, but it is analogous to a case that we did study.

2.3 Probability

Wandering the streets of Brisbane, just inside a dark alley you spot a man wearing a hat and sunglasses with a fold-up table in front of him. "Hey kid," he says, "wanna win some cash?"

He explains the game. He has a deck of 10 cards on the table in front of him, "but I'm not gonna tell you what's in the deck." You wager 20 dollars and then play 20 hands. Each hand is straightforward:

1. The man shuffles the cards.
2. The man offers his odds for black and for red. That is to say, he tells you the payout for each colour if you bet on that colour and win.
3. You bet either red or black.
4. You cut the deck.
5. The man turns over the top card.
6. If the card is the colour you bet on, then you win points equal to the offered odds for that colour, otherwise you win nothing.

After all 20 hands are played, the points are tallied and the man will pay out one dollar for each point (including cents).

After observing the man and how he plays the game you are confident of the following (i.e. you can take these as assumptions for the purposes of this exercise, but honestly don't play any game with a man in a back alley in real life!)

- There really are 10 cards in the deck and each card is either black or red, but you don't know how many of each there are.
- The process of shuffling, cutting the deck, and turning over the top card produces a uniformly random card from the deck.
- The man cannot know what card card will be drawn when he gives his odds, i.e. they are independent of the card drawn.
- The man doesn't really know what he is doing with the odds. They are independent of what cards are in the deck and change every hand. The odds are always at least 1 and at most 3.

Your goal is to write a Python function that makes bets to play the game with the man and maximise the expected winnings:

```
makeBet(blackOdds, redOdds, previousOutcome, state)
```

- `blackOdds` and `redOdds` are the points you earn when you bet on the corresponding colour and win.
- `previousOutcome` gives the outcome of the *previous* hand (either the string `'red'` or `'black'`), which you should use to update your models. On the first hand this will be `None`.
- Your function should return a pair: `(bet, state)`
- `bet` should be one of `'black'` or `'red'`, representing your bet

- `state` is a value of any type that your function returns and which is used for the `state` argument then next time your function is called. You can use this mechanism to allow your function to keep track of any information that you see fit. You should not use global variables to keep track of information as this may cause problems with the testing system.
- On the first hand, `state` and `previousOutcome` will be set to `None`.

Your Python program will be evaluated by playing the game 2000 times and calculating the average return (payout minus the 20 dollar wager for each hand). The distribution of black and red cards will be chosen uniformly at random for each game. Your mark on the Python program will be 1.5 times the average return, rounded up, capped at 15 marks. The best possible average return is around 10. So the awarded marks will be

$$\min(15, \lceil 1.5a \rceil)$$

where a is the average return or 0 if the average return is negative.

Please note that there is a time limit of 10 seconds for running the test. This should be plenty of time. Matt's slow solution runs in 3 seconds and his fast solution runs in 0.2 seconds.

3 Report

Your report should explain your solution method. Use mathematical language, concepts and notation from the unit to describe the problem and your solution. You should make use of mathematical tools and problems discussed in the unit such as regular expressions, finite state automata, matrices, vectors, Bayes rule, decision theory, etc.. Describe your implementation in Python. Mention the role of important variables, library calls (eg. to `numpy` or `re`, `probability.py` etc.), and source of reused code if applicable and any modifications required to it. If you need to make a choice about data structures, explain your choice.

Overall, your report should be understandable by another student in CAB203 who knows the material but hasn't thought about the task. It is not necessary to define terms already used in the unit, but you should point out the significance of particular details about the problem and the choices that you make.

Your report will be a single file, in PDF format. There is no maximum page length, but a concise, easy to understand report is better than a long wordy report. 1 or 2 pages is about right.

4 Python implementation

Your solution should be a reasonable implementation of the mathematical solution described in your report. The problems are all solvable using the Python concepts and syntax used in the unit. You can use additional syntax if you like. The marking system will use Python 3.10, so if you are using a later version be sure not to use any syntax newer than 3.10.

You are allowed to use or modify any functions defined throughout the lectures, tutorials, and assignment solutions. Many of these functions are collected in Python files `probability.py`. You can import these files rather than copying from them. You can assume that these files are available; there is no need to include them in your submission. Additionally, you are allowed to use the `csv`, `numpy`, `numpy.linalg` and `re` Python modules. Before using other modules, please contact the unit coordinator.

A submission template file for each task is available from the Canvas Assessments module. If your solution includes modified code from the unit, say so in a comment explaining where you obtained it and what modifications you made. One line is enough detail.

Test files are included to help you debug your code, available from the Canvas Assessments module. **Please make sure that you run the one for your task before submission.**

The `test_problemSolving_fsa.py` and `test_problemSolving_linalg.py` files can be used with Pytest or on their own. `test_problemSolving_probability.py` does not use Pytest and should be run with Python.

The tasks are designed so that they do not require long programs. Matt's longest solution is about 50 lines. There is no limit on the length of your program. However, the marking system will impose a time limit of about 10 seconds to avoid problems with infinite loops. This should be plenty of time to solve the tasks given.

Your code submission will be a single Python file. The filename doesn't matter, but it must end in `.py`.

Your Python code will be graded automatically by running test cases. The tests will be similar, but not identical, to those found in the test files. For the FSA and linear algebra tasks each test case is worth one mark. For the probability task, marks are awarded based on the average return as described in the task and implemented in `test_problemSolving_probability.py`, but with a different seed.

Your code is not assessed for quality, format, comments, length, etc.. Only the automated tests count for marks.

5 Generative AI options

For this assessment item we are trialling an option to use generative AI (eg. chatGPT) to prepare your report and Python implementation. You will have two options for how you want to do the assessment (AI or no AI) and each option has its own marking scheme, detailed in the next two sections.

Some things to consider:

- It is *optional* to use generative AI. There is no penalty for not using it, and no special bonus for using it.
- The grading will be different if you choose to use generative AI. There will be fewer marks awarded for the report and Python implementation, and you will need to write an additional report describing how you used generative AI (described in a later section below).
- Generative AI is not perfect. If you use generative AI then you will still need to check its work and polish its output. You are responsible for the quality of the report and code.
- If you choose not to take the generative AI option then the use of generative AI is not allowed for your submission.

Option	Report	Python	AI Report	Formula
No AI option	15	15	0	$r + p$
AI option	10	10	10	$\frac{2}{3}(r + p) + a$

Table 1: Marking breakdown for the two options. In the formula, r is the raw report mark, p is the raw Python code mark and a is the AI report mark.

The marking breakdown for both options is summarised in table 1.

You will declare your option through the submission processes:

- To choose the no AI option submit your report and Python code only.
- To choose the AI option submit your report, Python code and AI report.

5.1 No AI assistance option

With this option you are not allowed to use generative AI (eg. ChatGPT, copilot, and similar tools).

Your mark is made of two parts. The report is graded out of 15 and the Python code is graded out of 15, for a total of 30. Each mark counts 1% towards your final grade.

The marking rubric for the report will be made available on Canvas under Assignments > Problem Solving Report.

5.2 AI assisted option

With this option you are allowed to use generative AI (eg. ChatGPT, copilot, and similar tools).

Your submission will have a report and Python submission as described above, but will have an additional submission in the form of a second report describing how you used generative AI to produce the report and Python code. This AI report will include:

- Transcripts of your interactions with the generative AI including all prompts and responses or similar.
- A narrative of how you used the AI to solve the problem, including problems that you encountered, how you solved them, any modifications that you needed to make to the output, etc..
- URLs to the generative AI services and/or software that you used.

The AI report should be viewed as a description of the methods that you used. It should be understandable by a student from CAB203 who is not familiar with the AI tools that you used.

The AI report is worth 10 marks. The main report and Python code will be graded the same as in the no AI assistance option, but they will be scaled down to a maximum of 10 marks each, for a total of 30 possible marks for the assessment item.

The marking rubric will be made available on Canvas under Assignments > Problem Solving AI Report.

6 Submission

Submission process: You will need to make two submissions through two separate links in Canvas:

- Your report, in PDF format (extension `.pdf`).
- Your Python code, as a Python file (extension `.py`).
- If choosing the AI assistance option, your AI report, in PDF format (extension `.pdf`). If choosing the no AI option, do not submit an AI report.

You can find the submission pages on Canvas on the Assignments page.

Extensions: Information about extensions is available on the About Assessments module on Canvas. If you obtain an extension, please attach confirmation *as a separate file* to your report submission.

Citing your sources: You are welcome to source information and code from the internet or other sources. However, to avoid committing academic misconduct, you must cite any sources that you use. See https://www.citewrite.qut.edu.au/cite/ for guidelines on citing sources and how to properly format and acknowledge quoted material.

You are welcome to use resources, including code, from within the unit. Please cite the unit like *CAB203, Tutorial 7* or similar. This is only necessary when explicitly quoting unit material. There is no need, for example, to cite the definition of a graph or similar, unless you are directly quoting the lecture's definition.

For code, please include your citation as a comment within the code. For example

```
# modified from CAB203 graphs.py
```

Policy on collaboration: We encourage you to learn from your peers. However, for assessment you need to turn in your own work, and you will learn best if you have spent some time thinking about the problem for yourself before talking with others. For this reason, talking with other students about the project is encouraged, as long as you are putting in the effort on the problems yourself as well. But do not share your code or your report with other students and do not copy from others. It is considered academic misconduct to copy from other students or to provide your work to others for the purposes of copying.

For Slack and other online discussions, please do not post about solutions. Keep your discussions private so that everyone gets a chance to get to the solutions on their own. You can direct message or email the unit coordinator or one of the tutors if you wish to discuss specifics of your solution. Feel free, however, to ask general questions about the project on the Slack channels.