SUBMISSION

The submission consists of **2** files, compressed into a .zip or .rar file and uploaded into Canvas.

- A **document** in format .pdf (preferred) with the solutions and justifications to ALL the exercises. If you want to create documents by hand and scan/photograph them to add them to the pdf document, please ensure the quality is good and readable.
    - You must include a **link** to your shared **git repository** at the beginning of the document. The code should be self-contained; i.e., done in a way such as if anyone else wants to "download" your code and run it, they can. Assume everyone has the 4 basic libraries already installed (NumPy, Matplotlib, Pandas and NetworkX) – no need to reinstall those.
    - The document should also include the **code** for each exercise in a readable format. You can include the code after each exercise or all together at the end. Please label the code clearly.
- A **video**, preferably in format .mp4, no longer than 10 mins, with the explanation of your solutions for questions 1 and 2.

WEIGHTING

This submission is worth **60%** of your final mark. The deadline for submitting this is **Friday the 17th January 2025, at 6pm UK time**. Please allow for time to submit and take into account the different timezones if you are submitting from overseas.

As the final submission, in this assignment you will need to demonstrate your knowledge in all the content related to the module: Units 1, 2, 3, 4, 5 and 6.

The code must be your own; you might use functions of libraries such as "ones" or "randint" in NumPy, but the main code must be your own work.

TASKS

1. (25 marks) Code the Knight's tour in its both versions, open and closed. The "Open" version means that the knight can finish in a square that is not the start after passing through all the squares in the board. The "Closed" version means that the knight must finish its tour in the starting square. The instructions are as follows:

a. Save the chessboard (8x8) into an appropriate variable/data structure and display it in your terminal.

b. Ask the user whether they want an Open or Closed solution.

c. Ask the user what approach they want to use (Backtracking or Las Vegas).

b.i) When applying a Las Vegas approach, the end conditions are either that the tour is completed (successful) or the knight steps in an already-visited square (unsuccessful).

b.ii) For the Backtracking approach, you MUST explain in the report what is your rationale for choosing a particular direction and how you do the backtracking.

d. Give a visualization of all the visited "squares" (positions) when the program ends, regardless of if it is successful or not.

Compare both approaches and answer the following questions in your report:

a) What are their differences?
b) Which data structure/s would you use to implement this problem? Give your reasoning.
c) What is the success rate of each of the algorithms after 10000 runs?

Record yourself going over your code and explaining the rationale.

Hint: To know the success rate, record which runs give a successful result. Then, divide the successful number of runs over the total amount of runs.

2. (25 marks) Using the Python programming language, code one of the Minimum Spanning Tree (MST) algorithms, either Prim's or Kruskal's, for the following graph:

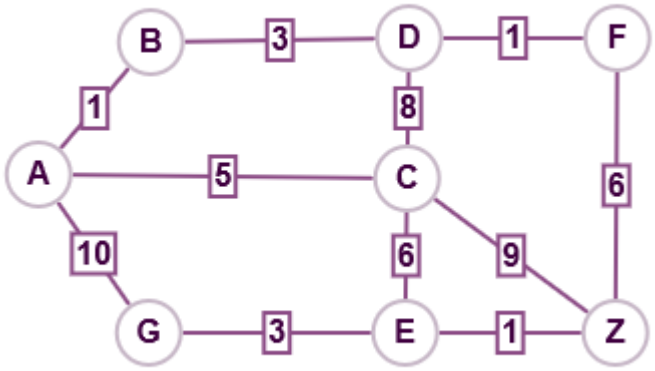

Figure 1: Graph

You must:

a. Depict the original graph first.
b. State which MST algorithm you are following.
c. Show the creation of the MST step by step.
d. Depict the final MST.

Record yourselves explaining the MST algorithm that you selected, citing your sources. Then, explain the rationale of your code and include a clip of your code running and building the MST step by step.

3. (20 marks) Provide the pseudocode for an in-place quicksort algorithm that sorts words alphabetically. The pivot must be always the **second-to-last** element in the array. Implement that pseudocode using Python.

Is there any other way to code a quicksort algorithm? Cite your sources to answer this question.

Hint: you can use auxiliar memory.

4. (10 marks) Palindrome.

A word is called a palindrome if it is read the same forwards and backwards. Examples of palindromes are "gag", "pop", "hannah" or "rotator".

You need to create a piece of code in Python that checks if a word is a palindrome or not using recursion. To solve this question you must implement a boolean recursive function called *isPalindrome()*.

5. (20 marks) Please solve the following string alignment problem using Needleman-Wunsch (no coding required). You need to use the optimal alignment algorithm with the strings GATATACC and ATACATA.

Please provide the alignment matrix, the traceback and the string alignment.

Remember that the penalty for both a substitution and a gap is -1. A match score is +1, but it needs to come from the diagonal.

MARKING SCHEME

The submission will be marked to a maximum of 100 marks. A breakdown of the marking scheme can be found below. There is a rubric available as well.

Exercise	Marks
Knight's tour	25
Minimum Spanning Tree	25
Sorting	20
Palindrome	10
Needleman-Wunsch	20