Technical Assessment: Industrial / Mechanical Simulation & Modelling Engineer

---

This technical assessment has been created to evaluate your competencies in object-oriented programming (OOP) with Python and your ability to implement computationally efficient solutions for real-world problems. It is important to remember that this is a test of effort and best practices - the answers you provide need not be complete or correct, but they must exhibit a demonstrable thought process.

Note:

- The use of generative AI (e.g., ChatGPT, GitHub Copilot, or similar tools) is strictly prohibited. To ensure compliance, the submitted code will be thoroughly reviewed for indications of AI assistance. Submissions found to be non-compliant will result in disqualification.
- Your solution will be evaluated based on its correctness, efficiency, and code quality. Additional credit will be given for demonstrating best practices in Python programming and clear documentation.
- If you install any packages or third-party libraries, please utilise a virtual environment and create a `requirements.txt` file - no assessment solutions will be reviewed without this file being present if needed.
- You will be required to verbally summarise and justify your solutions in the event that you are selected for further assessment with regards to the applied-for position.

Case Study 1 - Classes, Algorithms, & Testing

Problem 1:

You are given an array of `Component` instances representing nodes in a directed graph. Each of these instances has references to its downstream components via a unique identifier (a string value). This is visualised by the diagram accompanying this assessment.

Your goal is to determine the order in which to process these components such that the `compute()` instance method of each component is executed only after all its upstream components have been computed, in an effort to prevent downstream components from referencing empty / stale upstream instance variables.

Problem 2:

It is known that many of these component instances are to be created. Please implement a means to mitigate the memory constraints associated with this, whilst conforming to good object oriented Python practices. Additionally, elaborate upon the implications of this solution.

Hint: no additional methods, or means of component preprocessing, need be created. This solution requires you to 'adjust' how Python class key-value stores operate.

Problem 3:

Implement (in words) a method to compute subsets of this directed graph in parallel. In addition to this, explain your understanding of multiprocessing, and list some important considerations thereof.

Case Study 2 - Simulation, Sampling, & Modelling

Problem 1:

A simulation runtime uses a list of components as an input. Your goal is to create a sample space of simulation scenarios by parametrising some of the attributes of said components, given:

- The name of the attribute as a string.
- The upper and lower bound (within which the attribute's value must vary).

In the code file provided, these are presented by a dictionary called `sample_allocation`, which is to supercede the `parameters` dictionary.

Additionally, in an ideal case, you should allow the number of simulation scenarios to be specified, such that the ranges of attribute values do not dictate the size of the sample space. The goal is to parametrise a number of these components, and this should be accommodated in your solution.