**Overview**

The assessment is comprised of 3 short problems, described below.

**Evaluation criteria**

* Code implementation: style (clarity, type hints) & accuracy
* Ability to identify, debug, and resolve issues in an existing codebase
* Discussion of solutions

**Technologies recommended**

* Language - Python3
* Frameworks - Any
* Typehints - Mypy

**Problem #1 - Data Stream Ingestion**

In a data stream, data is read in consecutive chunks - so you only have access to a certain portion of data at a given time. This is in contrast to having an entire sequence available at once. For example, Netflix and Youtube use streaming to allow users to watch chunks of video, rather than have user wait for the entire video to load - before they can start watching.

Design a system that receives a stream of strings along with timestamps. Each **unique** string should be printed at most every **5 seconds** (i.e., string printed at timestamp t will prevent the same message from being printed until t + 5 seconds have passed).

All strings will be received in chronological order. Several strings may arrive at the same time.

Implement the DataStream class:

* + DataStream creates a data\_stream object
  + bool should\_output\_data\_str(int timestamp, str data\_str) returns true if the data\_string should be printed in the provided timestamp, otherwise returns false

**Input**

data\_stream = DataStream()

data\_stream.should\_output\_data\_str(timestamp=0, data\_string=“hello”)

data\_stream.should\_output\_data\_str(timestamp=1, data\_string=“world”)

data\_stream.should\_output\_data\_str(timestamp=6, data\_string=“hello”)

data\_stream.should\_output\_data\_str(timestamp=7, data\_string=“hello”)

data\_stream.should\_output\_data\_str(timestamp=8, data\_string=“world”)

**Output**

[true, true, true, false, true]

**Problem #2 - Fuel Station Design**

Design a fuel station for 3 types of vehicles - Diesel, Petrol, and Electric. There are a fixed number of spots for each type of vehicle at the fuel station.

Implement the FuelStation class:

* + FuelStation(int diesel, int petrol, int electric) creates a FuelStation object. The number of spots for each type of fuel is defined by the values provided to the constructor.
  + bool fuel\_vehicle(str fuel\_type)looks up whether there is an open slot that can provide fuel\_type. **A vehicle can only be fueled in a slot space of its**fuel\_type. If there is no slot free, return false, else put the vehicle in that fuel slot and return true.
  + bool open\_fuel\_slot(str fuel\_type)releases a fuel slot of fuel\_type so that another vehicle can be fueled. If you try to open a fuel slot, when all slots of a fuel\_type are empty, return false. Otherwise, return true.

**Input & Output:**

fuel\_station = FuelStation(diesel=2, petrol=2, electric=1)

fuel\_station.fuel\_vehicle(“diesel”) -> **true** (1 slot now open)

fuel\_station.fuel\_vehicle(“petrol”) -> **true** (1 slot now open)

fuel\_station.fuel\_vehicle(“diesel”) -> **true** (0 slots now open)

fuel\_station.fuel\_vehicle(“electric”) -> **true** (0 slots now open)

fuel\_station.fuel\_vehicle(“diesel”) -> **false** (0 slots open)

fuel\_station.open\_fuel\_slot(“diesel”) -> **true** (1 slot now open)

fuel\_station.fuel\_vehicle(“diesel”) -> **true** (0 slots now open)

fuel\_station.open\_fuel\_slot(“electric”) -> **true** (1 slot now open)

fuel\_station.open\_fuel\_slot(“electric”) -> **false** (only 1 slot available at fuel station)

**Problem #3 - Debug Calendar Design**

Your teammate has implemented a calendar program, but it doesn’t work as expected.

Ideally, a user should be able to add a new event, if it does not cause a double booking. Double bookings occur when two events overlap in time. For example, Event A lasts from Saturday 2 pm - 5 pm and Event B lasts Saturday 3 pm - 4 pm. These events overlap and would be considered a double booking.

Events are represented as a pair of ints - start and end. Two events can be scheduled back to back, e.g. Event A can be [2,3) and Event B can be [3, 4).

Your teammate has already implemented the Calendar class:

* Calendar() Initializes the calendar object.
* boolean schedule(int start, int end) Returns true if the event can be added without causing a double booking. Returns false otherwise and does not add the event to the calendar.

Your teammate’s code can be found in the attached file calendar.py. Please debug the code to fix the current issues. When submitting this portion of the assignment, please attach your fixed code and a README.md file that explains how you debugged it step-by-step and what the underlying issue was. We may discuss this document at the end of the evaluation.

**Example Input & Output**

calendar = Calendar()

calendar.book(5, 10) -> **Expect True**

calendar.book(8, 13) -> **Expect False**

calendar.book(10, 15) -> **Expect True**

**How to submit your work for assessment**

* Complete the assignment, one file per problem
* Push your code and supporting documents to your git account
* Share git url (public)

Disclaimer:

*This document is meant to assess your technical skills and is classified as "Infosoft*

*confidential". This document by any means shall not be used/shared without permission from Infosoft, non-adherence to this will considered as breach and can get your candidature blocked for employment with Infosoft.*