How to Run:

In each Terminal:

```
clear; foreman start -f Procfile.1-2
```

```
python watchdog.py
```

```
python tuplemanager.0.py 224.0.0.1 54324
```

```
python tuplemanager.1.py 224.0.0.1 54325
```

```
clear; foreman start -f Procfile.2
```

Problem 1:

- What the problem is: the system originally had a single point of failure on the tuplespace manager. So it was decided that we could stand up multiple copies of the tuplespace manager. However, there needed to be a single coordinator in case a new one started up or a current server went down.
- What triggers the problem, and how to reproduce it: the trigger was that there was only one server. Trying to stand up more than one would cause conflicts because each server would try to be the main server without any formal way to determine the leader.
- What impact the problem has on the system: having only one tuplespace is a single point of failure. However, we still need to have a way to elect a leader.
- What the correct behavior should have been: we needed a way to select a leader from multiple servers.
- Solution: Implement the Bully Election algorithm. Upon startup the server would call for an election and according to the algorithm, the highest id that responded would become the leader. The servers would also listen for election events and put their hat in as a candidate. In addition, there is a watchdog service that verifies that the leader is still alive. If not, the watchdog calls for an election of the remaining servers.

Problem 2:

- What the problem is: the system originally had a single point of failure on the tuplespace manager. So it was decided that we could stand up multiple copies of the tuplespace manager. However, there needs to be a way for the primary and backup servers to remain in sync.
- What triggers the problem, and how to reproduce it: the trigger was that there was only one server. Standing up multiple servers without any further changes would cause inconsistency between the primary and backup servers.
- What impact the problem has on the system: having only one tuplespace is a single point of failure. However, we still need to ensure that all the servers remain in sync.
- What the correct behavior should have been: we need a way for the primary and backup servers to remain in sync.
- Solution: Implement the Primary-Backup Replication. Once the server is elected leader, the leader ensures that anything that is written to it's tuplespace is also written to the tuplespace of the backups.