Done by: Tan Wei Jie (A0202017B)

Question 1, Part B:

Sequence of Events: (Seed 1234)

1. Shuttle1 (S1), Shuttle2 (S2), Shuttle3 (S3) and Shuttle4 (S4) are at these stations and waiting for orders:
    a. S1 – Station 1
    b. S2 – Station 1
    c. S3 – Station 2
    d. S4 – Station 3
2. The first order was accepted by the shuttle management system and broadcasted to all shuttles (S1, S2, S3, S4).
3. The shuttle replied with the following quotes:
    a. S1 replied with $8
    b. S2 is unable to accept
    c. S3 replied with $4
    d. S4 is unable to accept
4. The shuttle management system informed S3 to take the first order and rejected all other shuttles.
5. The second order was accepted by the shuttle management system and broadcasted to all shuttles (S1, S2, S3, S4).
6. The shuttle replied with the following quotes:
    a. S1 replied with $4
    b. S2 replied with $8
    c. S3 is unable to accept
    d. S4 replied with $6
7. The shuttle management system informed S1 to take the second order and rejected all other shuttles.
8. In the meantime, S3 is travelling on Track21.
9. S3 had reached Station 1 and picked up passengers.
10. S1 is travelling on Track12.
11. S3 is travelling on Track14.
12. S3 had reached Station 4.
13. S1 had reached Station 2 and picked up passengers.
14. S3 is travelling on Track43.
15. S3 had reached Station 3 and dropped off passengers.
16. S1 is travelling on Track23.
17. S1 had reached Station 3 and dropped off passengers.
18. S3 is stationed at Station 3 without load.
19. S1 is stationed at Station 3 without load.

Question 1, Part C:

Using the variable `totalTrainOnTracks` which increments when there is a shuttle travelling on the track, and `totalCustomerInSystem` which increment when a passenger boarded the shuttle.

Define all shuttles are at station equates to no shuttles are currently using the tracks
→ `#define no_train_on_track (totalTrainOnTracks == 0)`

Define all shuttles are without load equates to there are currently no passengers on any shuttle
→ `#define no_passenger (totalCustomerInSystem == 0)`

To check that the system always return to the above 2 states, the LTL property to check in SPIN is
`ltl v1 { []<> (no_passenger && no_train_on_track) }`

Another problem in this system is that orders that can be fulfilled may not be fulfilled. If the passenger requirement for all orders is below the maximum capacity of all shuttles, all orders can potentially be fulfilled.

Given this scenario:

- The first order: 5 people from station 1 to 3
- The second order: 5 people from station 2 to 3
- Same 4 shuttles given in Part A.

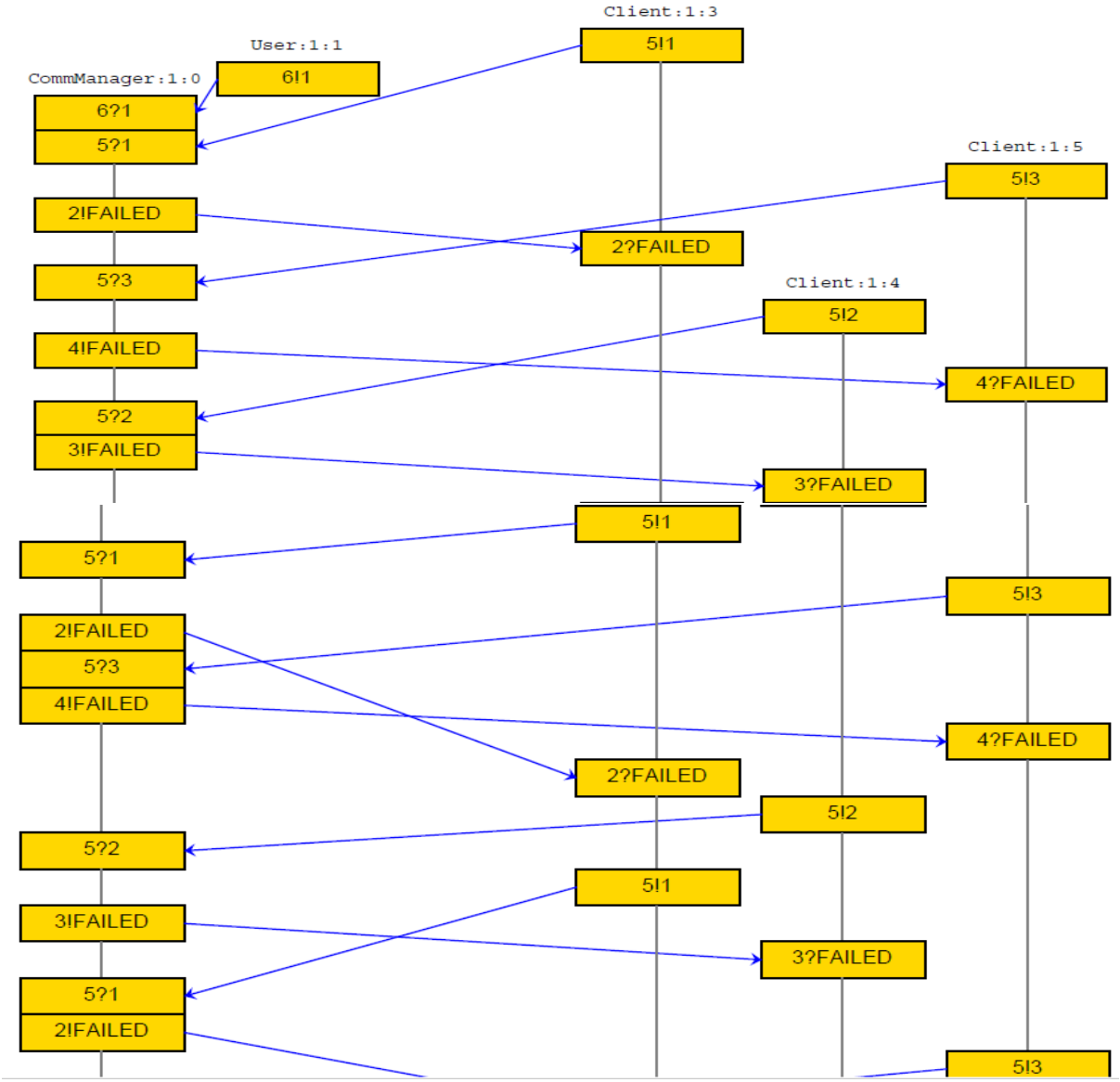
Define all jobs are fulfilled
→ `#define all_jobs_fulfilled ( job[1] == 1 && job[2] == 1 )`
→ `ltl v2 { []<> (all_jobs_fulfilled) }`

This LTL property will be violated in the current requirement because only 1 shuttle is capable of carrying 5 people and there are traces where while the first order is being carried out, the shuttle will reject the next order, and this rejected order will not be requested again.

Question 2, Part B:

A deadlock can occur when the communication manager is not idle, and disconnected clients try to repeatedly initiate a connecting request to the communication manager.

In the scenario below, the user had initiated a weather update, making the communication manager's status to "pre-updating". In the meantime, 3 disconnected clients tried to initiate a connecting request to the communication manager. The communication manager had to keep refusing the connection, and not be able to finish the weather update, thus causing a live lock.



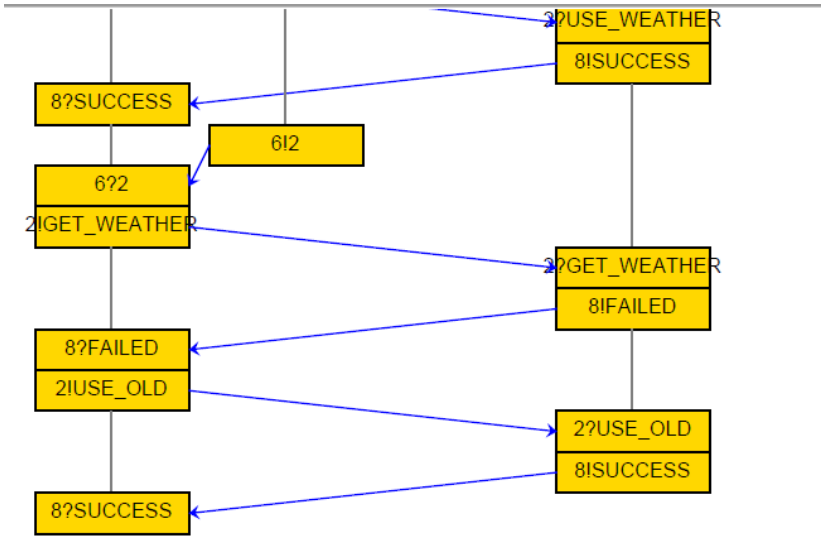
Another problem could be after post-reverting is completed successfully, the new weather status is not propagated again, causing a difference between the new weather update and the stale weather condition each client had.

LTL Property: All clients eventually use the new weather update

```
#define updatedWeather (clientWeather[1] == WCP_current_weather)
&& (clientWeather[2] == WCP_current_weather)
&& (clientWeather[3] == WCP_current_weather)
&& …

ltl v1 { []<> (updatedWeather) }
```

Using only 1 client and 1 weather update for this example. The user wanted to update the weather to 2, however, Client1's GET_WEATHER had failed, so it utilised the previous weather which is 1. There is no weather update after this and the program ended, thus, `WCP_current_weather != clientWeather[1]`.




```
WCP_current_weather  =  2
WCP_enabled  =  1
clientWeather[0]  =  0
clientWeather[1]  =  1
clientWeather[2]  =  0
clientWeather[3]  =  0
isConnected[0]  =  NO
isConnected[1]  =  YES
isConnected[2]  =  NO
isConnected[3]  =  NO
status[0]  =  IDLE
status[1]  =  IDLE
status[2]  =  IDLE
status[3]  =  IDLE
```