**Cohort Exercise 5**

**Name:** Kripashree S.

**Student ID:** 1006507

**Question 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Program Counter (line num)** | **Call Stack** | **Micro Queue** | **Promises** | **Macro Queue** | **Event Reg** | **Console Output** |
| 5 | [main()] |  | {promise1@5} |  |  |  |
| 8 | [main()] |  | {promise1@5, promise2@8} |  |  |  |
| 22 | [main()] |  | {promise1@5, promise2@8} |  | {ev1.run: function@22} |  |
| 26 | [main()] |  | {promise1@5, promise2@8} |  | {ev1.run: function@22, ev2.run: function@26} |  |
| 30 | [main()] |  | {promise1@5, promise2@8} |  | {ev1.run: function@22, ev2.run: function@26} |  |
| - | [main()] |  | {promise1@5, promise2@8} |  | {ev1.run: function@22, ev2.run: function@26} |  |
| - | [main(), ev2.run] |  | {promise1@5, promise2@8} |  | {ev1.run: function@22, ev2.run: function@26} |  |
| - | [main(), ev2.run] |  | {promise1@5, promise2@8} |  | {ev1.run: function@22, ev2.run: function@26} |  |
| - | [main(), ev2.run] |  | {promise1@5, promise2@8} |  | {ev1.run: function@22, ev2.run: function@26} |  |
| - | [main(), ev2.run] |  | {promise1@5, promise2@8} |  | {ev1.run: function@22, ev2.run: function@26} |  |

1. The run command on ‘ev2’ triggers the commands on line 26. It logs the message “data 1 received by ev2” into the console and then calls ‘foo’ with value 1. Inside ‘foo’, since 1 is an odd number, the code runs on ‘ev1’ with a value 2. The emission of ‘run’ on ‘ev1’ triggers the event listener defined on line 22. It logs the message “data 2 received by ev1” and proceeds to call ‘foo’ with the value of 2. Inside ‘foo’, since 2 is an even number, the code runs on ‘ev1’ with a value of 3. Since the cycle does not terminate and continues infinitely, there will be no output in the console.
2. If the program leads to a non-termination, the cycle will continue indefinitely with ‘ev1’ and ‘ev2’ running alternatively. The event loop will keep looping until all expected closures have returned and no more pending tasks remain in the micro or macro queues.