Node.js Assessment

💡 Problem Statement:

Amid the pandemic situation where traveling through public transport is still not considered a safe option, EPAM an organization that always promotes an employee first culture plans to provide **free cab services** for all its employees to commute to their office.
As a part of this service, the organization hired some cabs and the technical team at EPAM wants to build a simple automatic **Cab Allocation System**, that automatically manages allocation of cabs to the employees based on the availability of the cab pool.

📜 Required Representations:

- Represent an **employee pool** using a suitable data structure
- Represent a **cab pool** using a suitable data structure
- Represent **booking slots** using a suitable data structure - Morning(**0**), Afternoon(**1**), Evening(**2**), Night(**3**)
- Represent **request types** using a suitable data structure - Allocation request(**1**), Cancellation request(**-1**)
- Represent empty pool and on-going trips by **keeping the system idle/sleep** for required seconds
- To keep it simple, please assume all booking/cancellation **requests are made sequentially** (no parallel requests).

📐 Assumptions:

- **Employees** working in any shift **(morning / afternoon / evening / night)** can avail this facility for free of cost

⚠️ More than **half of the cab pool** cannot be allocated to the same slot.

- **No of employees** in the organization should always be **greater than** the **number of cabs** hired by the organization.

> *"no of employees => number of cabs"*

- A cab (if available) is **allocated soon after the request** without any delay. Successful allocation of a cab to the employee should **remove the cab from the available** cab pool.

> *"Request for cab => check availability and other constraints => accept the request & allocate cab (or) reject request*
> *If cab allocated => remove the cab from cab pool"*

ℹ️ An employee can book **only one slot at a time**. To make a new allocation request, the employee needs to complete/cancel the existing trip

ℹ️ When **no cabs** are available in the cab pool, system should **not accept any new requests.**

- An allocated cab **completes** its trip in **60** seconds. After completion of the trip the cab should be **added back to the available** car pool.

> *"On trip completion (60 seconds of trip) => add the cab back to the cab pool"*

- An employee can **cancel the booking** anytime (to keep it simple we assume that cancellation is possible even when the trip is on), any such cancellation will **add the cab back to the available** cab pool

> *"Booking cancelled by employee => add the cab back to the pool"*

⚠️ **Two consecutive cancellations** would **block** the employee from making further requests for a **cooling period** of 80 seconds.

- Cabs are usually allocated on a **FCFS**(First Come First Serve) basis. Any cancelled booking will be added to the end of the pool.

✏️ Test Cases:

Input Format:

- **First line** represent number of employees
- **Second line** represents number of cabs
- **Next consecutive set of lines** represents the cab requests in the format

```
<emp-id> <slot> <request/cancel>
Eg:
1 0 1  => <employee-1> <morning slot> <book cab>
4 2 -1 => <employee-4> <afternoon slot> <cancel cab>
```

Output Format:

Status (text) of each cab requests made above

TC #1: Cab allocations & handling empty pool
Input

```
5
3
1 0 1
1 1 1
1 0 -1
2 2 1
1 0 1
2 1 1
3 3 1
4 2 1
5 0 1
```

Expected output

```
Cab-1 allocated to Emp-1 for slot 0 (remaining cabs = 2)
Emp-1 is already allocated cab for slot 0 (remaining cabs = 2)
Booking for slot 0 cancelled by Emp-1 (remaining cabs = 3)
Cab-2 allocated to Emp-2 for slot-2 (remaining cabs = 2)
Cab-3 allocated to Emp-1 for slot-0 (remaining cabs = 1)
Cab-1 allocated to Emp-3 for slot-3 (remaining cabs = 0)
** No cabs in the pool. Please wait... **
** Cab-2 added back to the pool **
Cab-2 allocated to Emp-4 for slot-2 (remaining cabs = 0)
** No cabs in the pool. Please wait... **
** Cab-3 added back to the pool **
Cab-3 allocated to Emp-5 for slot-0 (remaining cabs = 0)
** Cab-1 added back to the pool **
** Cab-2 added back to the pool **
** Cab-3 added back to the pool **
```

TC #2: Block users on subsequent cancellations

Input

```
5
3
1 0 1
1 0 -1
1 1 1
1 1 -1
1 1 1
```

Expected Output

```
Cab-1 allocated to Emp-1 for slot 0 (remaining cabs = 2)
Booking for slot 0 cancelled by Emp-1 (remaining cabs = 3)
Cab-1 allocated to Emp-1 for slot 1 (remaining cabs = 2)
Booking for slot 1 cancelled by Emp-1 (remaining cabs = 3)
** Sorry request denied. Please wait for 80 seconds and try again **
Cab-1 allocated to Emp-1 for slot 1 (remaining cabs = 2)
** Cab-1 added back to the pool **
```

TC #3: Not more than half of the cab pool can be allocated to the same slot

Input

```
10
5
1 0 1
2 0 1
3 0 1
4 0 1
4 1 1
5 1 1
6 1 1
7 1 1
```

Expected Output

```
Cab-1 allocated to Emp-1 for slot 0 (remaining cabs = 4)
Cab-2 allocated to Emp-2 for slot 0 (remaining cabs = 3)
Cab-3 allocated to Emp-3 for slot 0 (remaining cabs = 2)
** Sorry cannot allocate for slot 0 at the moment. Please check
different slot or try later **
Cab-4 allocated to Emp-4 for slot 1 (remaining cabs = 1)
Cab-5 allocated to Emp-5 for slot 1 (remaining cabs = 0)
** No cabs in the pool. Please wait... **
** Cab-1 added back to the pool **
Cab-1 allocated to Emp-6 for slot 1 (remaining cabs = 0)
** Sorry cannot allocate for slot 1 at the moment. Please check
different slot or try later **
** Cab-2 added back to the pool **
** Cab-3 added back to the pool **
** Cab-4 added back to the pool **
** Cab-5 added back to the pool **
** Cab-1 added back to the pool **
```