**Case Study: Call Center**

A call center consists of below members

* ***X*** Junior executives at level 1 who attends the call first.
* ***Y*** Senior executives at level 2 who attends calls escalated from Level 1
* 1 Manager who attends the calls escalated from Level 2
* **X > Y** always
* Call center receives ***Z*** calls a day

Calls should be assigned to the executives in such a way that the total time taken to finish ALL the calls is optimized.

**System Rules**

* Any calls which stays more than 7 minutes with a JE will be escalated to SE
* Any calls which stays more than 10 minutes with a SE will be escalated to next level (Manager)
* Any calls which stays more than 15 minutes with Manager will be treated as *unresolved*
* Any executive cannot not handle more than **Z**/**X+Y** calls. Once the limit is reached, he/she should not receive any further calls.

**Sample Request Json :**

**{**

"number\_of\_calls": "30",

"je": **[**

"5,7,6,4,6",

"5,8,7,5,10",

"7,5,6,14,6",

"10,4,9,5,12",

"6,10,11,4,6"

**]**,

"se": **[**

"6,14,12,10,5",

"18,8,6,4,12",

"8,6,13,7,1"

**]**,

"mgr": "20,12,25,13,20,3,3,3,9,2,7,1,7,11,10"

**}**

**Where:**

The number of elements in the **je and se list denotes the X and Y respectively.**

**First element in the list above (je, se) , denotes the time taken by first executive, the second element denotes the time taken by second executive and so on..**

**For example, "** **9,12,15,13,6,3,3,3 is a mock of the time taken by the first Junior Executive (e.g. he would have taken 9 mins on first call, 12 mins for second etc.) …. Example shows 20 values against manager, mocking the worst case where all calls come to manager. In real life, these values would be the exact times taken by executives but this mock data is provided to simulate executive’s behavior in the program.**

***Expectations***: -

1) Implement as a **Micro service** using **TDD** approach, covering all the edge cases scenarios

2) Using a suitable design pattern is highly recommended. Also, design should be efficient to handle modifications on the go, ex: increasing/decreasing number of executives, etc..

3) Exception handling

4) The system should be capable of finding the **optimized** way to handle the calls with available resources whose details are provided in the request.

**5) Response should look like:**

**Sample Response JSON :**

**{**

"number\_of\_calls": "30",

"resolved": "23",

"unresolved": "7",

"totalTimeTakenInMinutes": "${timeTaken}",

"performance": [

{

"manager": {

"id": "mgr",

"timeTakenInMinutes": "57",

"callsAttended": "3",

"resolved": "1",

"unresolved": "2"

},

"junior-executives": **[**

{

"id": "je1",

"timeTakenInMinutes": "28",

"callsAttended": "5",

"resolved": "5"s

"escalated": "0"

},

{

"id": "je2",

"timeTakenInMinutes": "36",

"callsAttended": "5",

"resolved": "3",

"escalated": "2"

},

{

"id": "je3",

"timeTakenInMinutes": "38",

"callsAttended": "5",

"resolved": "4",

"escalated": "1"

},

{

"id": "je4",

"timeTakenInMinutes": "40",

"callsAttended": "5",

"resolved": "2",

"escalated": "3"

},

{

"id": "je5",

"timeTakenInMinutes": "37",

"callsAttended": "5",

"resolved": "3",

"escalated": "2"

}

**]**,

"senior-executives": **[**

{

"id": "se1",

"timeTakenInMinutes": "32",

"callsAttended": "3",

"resolved": "1",

"escalated": "2"

},

{

"id": "se2",

"timeTakenInMinutes": "30",

"callsAttended": "3",

"resolved": "2",

"escalated": "1"

},

{

"id": "se3",

"timeTakenInMinutes": "14",

"callsAttended": "2",

"resolved": "2",

"escalated": "0"

}

**]**

}

]

**}**

**Note: - Consider this as a contract on how the response should look like. If you find any mismatch between request & response, kindly ignore and focus on the rules explained in the problem**