Graduate / Intern coding test

Welcome to the Reckon coding test!

Build the code as if this was just one of many tasks that you have to do in one day, using whatever language / libraries / frameworks you feel comfortable with unless explicitly instructed not to. The tasks are fairly straightforward and should take no longer than an hour, there are no major "gotchas" but there will be input and output that exist beyond the samples provided. When finished, please create a repository and push the unzipped, raw source to a public bitbucket / github repository and then submit a link to this repository.

Please submit working source code to solve the problem along with any supporting code that you might have used in testing.

Test 1:

We have two api endpoints that provide a simple json response.

Endpoint 1:

https://join.reckon.com/test1/rangeInfo provides an upper and lower bound of numbers

An example response may be

```
{
    "lower": 1,
    "upper": 100
}
```

`Endpoint 2:`

https://join.reckon.com/test1/divisorInfo provides a list of outputs that we want to check for .

```
{
    "outputDetails": [
        {
            "divisor": 3,
            "output": "Boss"
        },
        {
            "divisor": 5,
            "output": "Hogg"
        }
    ]
}
```

Requirements

Given the two endpoints provided, provide working code that

- calls *Endpoint1* to get the upper and lower bound of numbers
- calls *Endpoint2* to get a set of divisors and the output required.
- For all numbers inclusive of the lower and upper bound, go through and check if it divisible by each of the divisors provided in *Endpoint2,* If the number is wholly divisible , log the output as a result. If multiple outputs are satisfied, print outputs that are satisfied.

Importantly

- Out api unreliable* and will sometimes not work, for extra marks if you have time, do what you can to build some resilience into your solution when calling our api. (* it's not really unreliable, we just deliberately made it that way)

Example Guideline

We have stipulated that any language choice or framework is open to you, but in case you want some guidance. You could build (but are not limited to):

- a nodeJS server side app, using a framework like express that sends back the result via the browser
- a nodeJS api endpoint that can be called and sends back a json payload
- a console .net / java app that prints out the result to the console
- a python script.
- a react front end that calls these api endpoints and shows the result in the browser.
- ...whatever you want.

Expected Output

Using the example output from the two endpoints above, we would expect to see the output:

Sample Output

```
1:
2:
3: Boss
4:
5: Hogg
6: Boss
7:
8:
9: Boss
10: Hogg
11:
12: Boss
13: Boss
14:
15: BossHogg

[....etc....]
```