**Generators Exercise:**

The following github repository is implementing a server which tries to send 3 requests to a different server, receives a number and aggregates the results:

<https://github.com/tamarstern/requestAggregatorNoPromises>

The aggregation is done with async parallel.

Pay attention that 2 tests fail –

* Both of the tests that fail mock an error received from the url we are sending requests to. Async parallel, while receiving an error, stops the execution of the tasks immediately. In that case, the server will stop to aggregate the results and will return the aggregated number he had collected so far.

Your Task is:

1. Re-write the async parallel to use generators – make sure all tests pass as your success criteria. The server should behave as follows :
   1. send the requests one after another
   2. aggregate the results
   3. Handle errors – if one of the requests fail – continue aggregate the results from the other requests.
2. Add the number of requests send as a parameter for the request and re-write the generator to receive a number N and send N requests.

Attention – Your code should look as much as possible like synchronous code. The goal of this exercise is to generate a code that looks as synchronous as possible – in the aspects of syntax and error handling – but will be asynchronous behind the scenes.