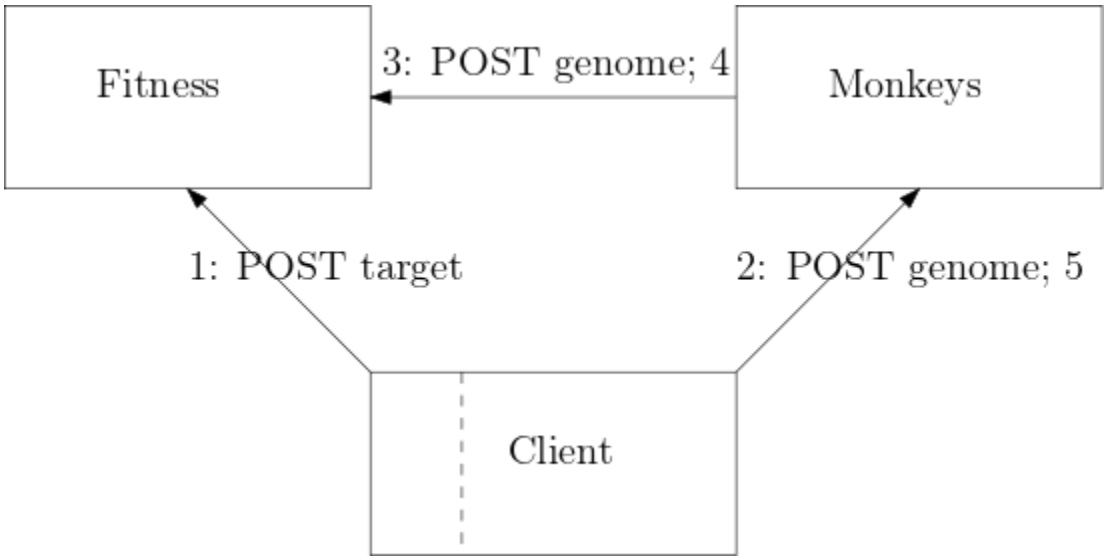
Assignment #2.2 – Hamming REST for Shakespearean Monkeys

Specs

The **current assignment**, **A#2.2**, is a smaller preliminary step, for the more following and substantial assignment, A#2.3. Essentially, the current A#2.2 prepares the required servers and their connections.

We use **ASP.NET**, but to avoid needless complexities, we use the upper open source layer known as **Carter**, which features quite a few functional elements.

We set up two standalone REST Carter servers: (1) **Monkeys**, listening on http 8081 and https:8082; and (2) **Fitness**, listening on http 8091 and https:8092. Here, the client will be simulated by testing scripts using **curl** and **httprepl**.



With the two servers running (on localhost):

1. **Client** posts a target text to **Fitness …/target** (empty ok response)
2. **Client** posts a genome (candidate) text to **Monkeys …/try**
3. **Monkeys** posts the received genome text to **Fitness …/assess**
4. **Fitness** responds to **Monkeys** with the **Hamming** distance between its stored target text and the received genome text
5. **Monkeys** returns this integer in his response to **Client**

Note

If strings have unequal lengths, then the **Hamming** distance is first computed for their minimum length, and then added with their lengths difference.

Sample scenario

- Target text, sent by the **client** to **Fitness** (step 1):

  `[To be or not to be, that is the question]`

- Candidate genome text, sent by the client to **Monkeys** (step 2), then forwarded to **Fitness** (step 3):

  `[To be~orAnoa [OBbej tVat i.Xt<eLju(s2ion]`

- Hamming distance, computed and returned by **Fitness** (step 4), and finally returned by **Monkeys** to **client** (step 5): **15**

More details for this scenario:

1. Client posts a **target** text to **Fitness …/target** (empty response)

Client = you (after donning the tester hat), or the marker, using one of the testing tools, such as **curl**, httprepl, or Postman. Pseudocode (not real code):

```
POST http:/localhost:8081/target
     {"text": "To be or not to be, that is the question"}
```

2. Client posts a genome (candidate) text to **Monkeys …/try**

Same client (you, marker), similar tools; pseudocode:

```
POST http:/localhost:8091/try
     {"text": "To be~orAnoa [OBbej tVat i.Xt<eLju(s2ion"}
```

3. Monkeys forwards the received genome text to **Fitness …/assess**

Thus, your **Monkeys** service acts now as client to **Fitness**; pseudocode:

```
POST http:/localhost:8081/assess
     {"text": "To be~orAnoa [OBbej tVat i.Xt<eLju(s2ion"}
```

4. Fitness responds to **Monkeys** with the **Hamming** distance between its stored target text and the received genome text. I.e., your **Fitness** service computes and sends this as its response to (**3**); pseudocode:

```
{"number": 15}
```

5. Monkeys returns this integer in his response to **Client**. I.e., your **Monkeys** service sends this as its response to (**1**); pseudocode:

```
{"number": 15}
```

Sequence 2,3,4,5 can be repeated any number of times, typically with other "genome" texts (but repetitions cannot be excluded).

Then, the whole **sequence 1,2,3,4,5** can also be repeated, typically with another target text (but repetitions cannot be excluded).

The markers will only post correctly formatted data, as in (**1**) and (**2**); so, we do not expect exceptions here.

JSON formats summary

```
{"text": "..."}
{"number": 15}
```

Submission

Submit two complete files, one for each server, called: **upi-fitness**, and **upi-monkeys**, where upi is your own upi, e.g. jbon007. The files must include **all** .cs code required to compile and run these services (including Program, Startup, and Module).

The markers will test your servers with simple scripts, e.g. based on **curl** (you do not have to worry about these).

Due date: Monday 5 October, 23:00, to ADB