**Objective:**

Using Node.js and MongoDB, create a RESTful web service to solve the snail problem.

Please check the “additional points” and ““what we are evaluating” sections to get a better score in the evaluation.

**The Problem**

A snail is at the bottom of a 6-foot well and wants to climb to the top. The snail can climb 3 feet while the sun is up, but slides down 1 foot at night while sleeping. The snail has a fatigue factor of 10%, which means that on each successive day the snail climbs 10% \* 3 = 0.3 feet less than it did the previous day. (The distance lost to fatigue is always 10% of the *first* day's climbing distance.) On what day does the snail leave the well, *i.e.*, what is the first day during which the snail's height *exceeds* 6 feet? (A day consists of a period of sunlight followed by a period of darkness.) As you can see from the following table, the snail leaves the well during the third day.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Day | Initial Height | Distance Climbed | Height After Climbing | Height After Sliding |
| 1 | 0' | 3' | 3' | 2' |
| 2 | 2' | 2.7' | 4.7' | 3.7' |
| 3 | 3.7' | 2.4' | 6.1' | - |

Your job is to solve this problem in general. Depending on the parameters of the problem, the snail will eventually either leave the well or slide back to the bottom of the well. (In other words, the snail's height will exceed the height of the well or become negative.) You must find out which happens first and on what day.

**Web Service Method**

Develop a RESTful node.js webservice with the resource Snail and it will take 4 parameters: H, U, D, F

The parameters H, U, D and F are integer data types, and the result should be output as a JSON encoded string.

This method should solve the Snail problem and then log all values in a MongoDB collection.

The “snail log” should be stored in a collection in the following format :

{

\_id: <ObjectId>,

h: <ISODate>,

h: <NumberDecimal>

u: <NumberDecimal>,

d: <NumberDecimal>,

f: <NumberDecimal>,

result: <String>

}

**Web Service Input**

Each web service call should contain four integers *H*, *U*, *D*, and *F*. All four numbers will be between 1 and 100, inclusive. *H* is the height of the well in feet, *U* is the distance in feet that the snail can climb during the day, *D* is the distance in feet that the snail slides down during the night, and *F* is the fatigue factor expressed as a percentage. The snail *never* climbs a negative distance. If the fatigue factor drops the snail's climbing distance below zero, the snail does not climb at all that day. Regardless of how far the snail climbed, it always slides *D* feet at night.

**Web Service Output**

For each call, the webservice should return a result. The result should be a JSON encoded string indicating whether the snail succeeded (left the well) or failed (slid back to the bottom) and on what day. Format the output *exactly* as shown in the example.

**Sample Webservices Call**

Each row in this table means an individual call to the webservice, the H, U, D and F are the input parameters for method

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **H** | **U** | **D** | **F** | **Result** |
| 6 | 3 | 1 | 10 | success on day 3 |
| 10 | 2 | 1 | 50 | failure on day 4 |
| 50 | 5 | 3 | 14 | failure on day 7 |
| 50 | 6 | 4 | 1 | failure on day 68 |
| 50 | 6 | 3 | 1 | success on day 20 |
| 1 | 1 | 1 | 1 | failure on day 2 |

**Deliverable:**

The deliverable for this Programming Test are all files needed to execute both the front end (client) and the back end (server), please send all your files by email to the person who sent you this test

**What we are evaluating:**

Since this is a test you are doing at home, you are allowed to consult any resource on the internet, and also review past projects. Basically there are no limits in what are you able to use to solve this test.

But we will evaluate the program based on the following criteria:

* The snail problem must be solved correctly according to test cases we provided.
* Using RESTful web services to solve the problem
* Check the section “additional points” to improve your final score.
* Clean source code and use of any CODE STYLE format.
* Maintainable source code. (comments, good separation of presentation layer from business logic layer)
* Speed. If you sent your project before deadline, you will gain additional points also.

**Additional points:**

Since this is a mid-level test and we want to get a better idea of your technical capabilities and skills, if you are able to, please complete the additional points listed below:

* Use a REST webservice in the backend, and allow the user to login via OAuth 2.0.
* Save all calls to the webservice in a MongoDB table, and provide an interface in the front end to review past executions.
* Aggregate data from past executions at a /report endpoint to return the total number of successes, failures, average total distance climbed, average time for success, average time for failure
* Write tests for your code using Mocha
* Use the Express framework