Oursky Developer Pre-Test 2023

Thank you for applying to Oursky. When considering applicants for developer positions, we focus on their technical skills, which can't be measured by their CV alone; That's why we hope you will take the time to complete this pre-test.

Please do not disclose the content of this test to others.

For the answers, please pick a programming language you're familiar with.

To answer the questions, **please create a gist link at gist.github.com, with 3 files named answer[1-3].md** (or the correct extension for the programming language, e.g., answer1.c / answer2.md / answer3.py), and **reply to our pre-test interview email with the gist link**. The gist link is expected to be configured as **non-searchable**. Also, please don't send a Github repo or zip file to us.

All 3 questions are mandatory.

Your answer will be evaluated based on:

- Correctness of your solutions
- Cleanness, expressiveness and clarity of your code and logic
- Simplicity and efficiency of the code

Please be reminded correctness of solutions is not the only evaluation criteria, especially for experienced candidates. Some common problems include misunderstanding the questions, failure to consider edge cases of the problems, and obvious redundant logic or data structures.

Question 1

Given a dictionary of variable names where each name follows CamelCase notation, print the item containing most words. Please note, abbr, like `ID`, `IP` are considered one word. i.e. name `IPAddress` contains two words, `IP` and `Address`.

If the name contains the same number of words, it output the longest and the first appearance.

Example:

```
Dictionary=["Hi","Hello","HelloWorld", "HiWorld", "HelloWorldWideWeb", "HelloWWW"]
Output="HelloWorldWideWeb"
```

```
Dictionary=["Oursky","OurSky","OurskyLimited", "OurskyHK", "SkymakersDigitalLTD", "Skymake
rsDigitalLtd"]
Output="SkymakersDigitalLTD"
```

Question 2

Design and implement a data structure for a cache, which has the following functions:

- `get(key)`

  Return the value associated with the specified `key` if it exists in the cache, else return `-1`.

- `put(key, value, weight)`

  Associate `value` with `key` in the cache, such that `value` might be later retrieved by `get(key)`.

  Cache has a fixed capacity, and when such capacity is reached, `key` with the least score must be invalidated until the number of `key`s falls within cache capacity. The score is calculated as follows:

  - `weight ／ [ln(current_time - last_accessed_time + 1) + 1]`

Implementation of the cache should be optimized on the time complexity of `get(key)`. For example, the average time complexity of `get(key)` could be constant.

For the purpose of implementing the cache, you could assume that common data structures, such as arrays, different types of lists, and hash tables, are available.

At the end of the answer, give and explain the computational complexity of `get(key)` and `put(...)` in Big O notation.

Question 3

Please understand the following program:

```
function recur(n, cur) {
  if (!cur) {
      cur = 0;
  }
  if (n < 2) {
    throw new Error('Invalid input');
  }
  if (n === 2) {
    return 1 / n + cur;
  }
  return recur(n - 1, cur + 1 / (n * (n -1)));
}
```

To prevent an infinite loop in a production system. Write a program doing the same calculation without recursion. Please be reminded that a while loop is also considered not good in a production system.