Coding Challenge

Hi There,

This coding challenge is one part of our recruiting process where you can show us your best coding and problem solving skills.

Developers at Stillfront Germany enjoy a lot of freedom; we take ownership of entire features from design to implementation to testing and deployment. We don’t tell you how to do things, because we assume you know best and you’ll do your best.

In this challenge, you’ll be creating an actual feature for Goodgame Empire. We chose this example to give you a brief insight on the types of problems we’re trying to solve.

The Problem

Every player in Empire has a castle. To let players attack another player’s castle, we need you to create armies of randomly distributed troops (a troop is a formation of soldiers with the same skill, such as Spearmen, Swordsmen, Archers, etc.).

For example, we'll call your code telling it we need a random army that's 167 men strong. Assuming our available unit types to be, for example, Spearmen, Swordsmen and Archers, what we want from you is that you tell us what such a random army would look like, e.g.

Our Input: 167

Example result:

63 Spearmen

57 Swordsmen

47 Archers

(The "text output" above is just an example; we need this as structured data)

Boundaries

- Each troop MUST be > 0
- The result MUST be different (non-deterministic) with each call. When we call your code 100 times with the same parameters, we expect 100 different results.
- Obvious biases in the result are strongly discouraged. An obvious bias would be if e.g. one troop type always is the largest (or smallest etc.)
- Downtime is the enemy. Once deployed, code updates are strongly discouraged. We want this to be deployed-and-forget while still being future proof.
- We appreciate an O(1) solution[1]

---

[1] O(1) describes an algorithm that will always execute in the same time (or space) regardless of the size of the input data set.

Remarks

- This is your solution and you own it end to end. That is, you are free to decide the design, implementation, docs, etc. Just give us your best.
- Please write this in PHP.
- Invest a reasonable amount of time, no more and no less.
- Your code must be correct and it is up to you to proof that it is actually working as intended (docs, testing and code coverage).
- We appreciate a smart algorithm. We'll look at your general way of problem solving, how elegant your solution is, that is, how well your solution performs with as few resources as possible (resources including CPU, memory, dependencies).
- We are looking for flexible solutions that can be easily extended and enhanced for future use cases.
- We also appreciate a good build and deployment setup.
- If you have questions please feel free to ask any time.

Sending your solution

Once you are done with your solution, please zip it and e-mail (margarita.kremhoeller@stillfront.com) it back to us.

If you have a demo running somewhere publicly, please include the URL. Essentially include everything that we need to run your solution.

Instructions are helpful.

Thank you very much in advance for investing your time in solving our coding challenge, we really appreciate your efforts.


Stillfront