R·I·T

Kenneth LA MARCA <kcl4048@g.rit.edu>

Campspot Developer Programming Challenge

Joshua Traxler | Campspot <careers+f30vc65j@campspot.recruiterbox.com> Tue, Mar 28, 2017 at 10:30 AM
Reply-To: careers+f30vc65j@campspot.recruiterbox.com
To: kcl4048@rit.edu

Kenneth,

Thanks for your time yesterday. We're pleased to let you know we're moving you to the next stage of our hiring process. At this stage we'd like you to complete the programming challenge described below. In addition, please send us 2-3 references.

We look forward to reviewing your solution.

Thanks,
Josh

==========
Campspot Programming Challenge
==========

Most destination resorts don't host guests who are willing to stay for only one night. Their guests are not simply passing through the area; they've often travelled far with lots of luggage specifically to stay there, and it's not worth it for them to stay one night. As a result, a reservation system for resorts needs to prevent one night gaps in its reservation grid, or schedule of inventory. If one-night gaps occur on the grid, they will not be sold, and this results in lost income for the resort owner. For example, Campspot uses a one-night gap rule, as this is a common problem for camping resort owners. However, owners' gap problems aren't limited to one-night gaps. Some resorts only host guests who stay a minimum three or four nights. Those resorts may want to prevent one-, two-, and three-night gaps.

To support this use case, Campspot has a configurable business rule called a "gap rule" that allows resort owners to define which types of gaps are not allowed.

Your task is to implement the most basic version of a gap rule, which prevents new reservations on a campsite that will create gaps of a specified size.

Deliverables we expect from you include:

1. Souce code for executable program that takes in a standard JSON input file (described below) and returns which campsites are available to book
2. Executable test cases for your program
3. Written documentation that explains:
   a. How to build and run your program and tests
   b. A high-level description of your approach to solving the problem
   c. Any assumptions or special considerations that were made

---

We've provided a sample input file (attached to this email), which you can use to test your program.

The expected output of the sample file is:
"Daniel Boone Bungalow"
"Teddy Roosevelt Tent Site"
"Bear Grylls Cozy Cave"
"Wyatt Earp Corral"

The JSON input file contains a single object with the following keys:

"search": the dates of the reservation we are hoping to make
"gapRules": an array of different gap rules. Each rule has a single property "gapSize" which is the size of the disallowed gap.
"campsites": An array of the campsites that are available to book, each has an id and name.
"reservations": An array of the existing reservations that must be considered. Each has a campsite ID, start date, and end date

---

Evaluation and Delivery:

Things we will be looking at when evaluating your submission:
1. Does your code work according to the provided specifications
2. Is your documentation clear and insightful
3. Is your code well structured and maintainable
4. Are your tests well structured, robust, and maintainable

Some questions we often ask ourselves when looking at submissions:
1. Does this code demonstrate an attention to detail?
2. How easy would it be to extend this code to add additional features?
3. How easy would it be to change any assumptions you made about the problem?
4. How easy would it be to plug this code into a different interface like a web service?

We recommend choosing a tech stack that you are experienced and comfortable with. We want you to impress us with you problem solving skills rather than your ability to learn new technologies.

Please deliver your solution as a git repository. Once we recieve and evaluate your solution, we may reach out to you to follow up with questions about your implementation.

---

test-case.json
3K