Software Assessment - Work Orders

You're a newly-hired software developer working at a home repair services company. This company uses a very old job scheduling system: plain text files that list all of their data models.

As part of their effort to join the 21st century, the company is transitioning to a more modern system. However, all of the previous data still exists as plain text files and will have to be converted. This is where you come in.

Your boss needs you to develop a utility that takes a pair of plain text files as the input (*orders* and *dependencies*) and outputs a structured JSON file that shows all work orders and their dependencies. The details are described in the sections below.

If you notice something is not working (like the API, or any of the links in this document), please contact hello@hatchways.io.

Evaluation

This assessment will be evaluated based on the following criteria:

- Correctness: Is your solution complete and does it pass different test cases?
- Code Organization, Readability, & Maintainability: Is your code easy to read and well organized?
- Code Performance: Is your code efficient? Did you use appropriate data structures?
- Best Practices: Did you utilize good programming practices (write unit tests, avoid anti-patterns)? Did you show a good grasp of your language/framework of choice?
- Completion speed: A fast completion time comparable to the completeness of your solution. This is the least important criteria.

We use the following rubric to evaluate your submission. **Please note that if your submission does not attempt to complete all of the requirements, we will be unable to provide feedback on it.**

Inputs

The inputs to your application are two text files. Here is a compressed folder with an example of the two input files. The text files are described in the sections below.

orders.txt

This file contains information about the orders. The file has the following columns:

- **id** - the id of the order (any integer between 0 and 10000)
- **name** - the name of the order (any string of length 1 to 100)

Here is the content of the example input:

```
id,name
1,Pick up pipes and tiles
2,Install tiles
3,Install pipes
4,Waterproof pipes
5,Remove old tiles
6,Rustproof pipes
```

dependencies.txt

This file contains all dependency relationships. Each row represents one dependency (i.e. in the example below, order 2 depends on order 1 being completed). The file has the following columns:

- **id** - the id of an order
- **child_id** - the id of an order that depends on the order in column 1

Here is the content of the example input:

```
id,child_id
1,2
1,3
3,4
5,2
3,6
```

Output

The expected output of the program is **a JSON file** that shows all the work orders and their dependencies.

Note:

- **At the root (first) level, only the orders that are not dependent on any other order are outputted**

- For each order, show all the dependencies for that order
- Orders can be shown multiple times (i.e they may depend on multiple orders)

Below is an example output given the example inputs


```
{
  "orders": [
    {
      "id": 1,
      "name": "Pick up pipes and tiles",
      "dependencies": [
        {
          "id": 2,
          "name": "Install tiles",
          "dependencies": []
        },
        {
          "id": 3,
          "name": "Install pipes",
          "dependencies": [
            {
              "id": 4,
              "name": "Waterproof pipes",
              "dependencies": []
            },
            {
              "id": 6,
              "name": "Rustproof pipes",
              "dependencies": []
            }
          ]
        }
      ]
    },
    {
      "id": 5,
      "name": "Remove old tiles",
      "dependencies": [
        {
          "id": 2,
          "name": "Install tiles",
          "dependencies": []
        }
      ]
    }
  ]
}
```


And here is another set of inputs and output.

Important: Make sure the output follows the format above (e.g, it's an object with key "orders") and that it is valid JSON (you can use a validator).

Testing (Bonus)

An important part of development is testing. As a bonus, you can add some unit tests to ensure your solution works for different cases.

Usage

Since your program will be used with different inputs, it must not hardcode the filenames or paths. Instead, take them as command line arguments. Arguments will be passed in the following order:

{path-to-orders-file} {path-to-dependencies-file} {path-to-output-file}

For example:
"python main.py orders.txt dependencies.txt output.json"
"java Main orders.txt dependencies.txt output.json"
"node app.js orders.txt dependencies.txt output.json"

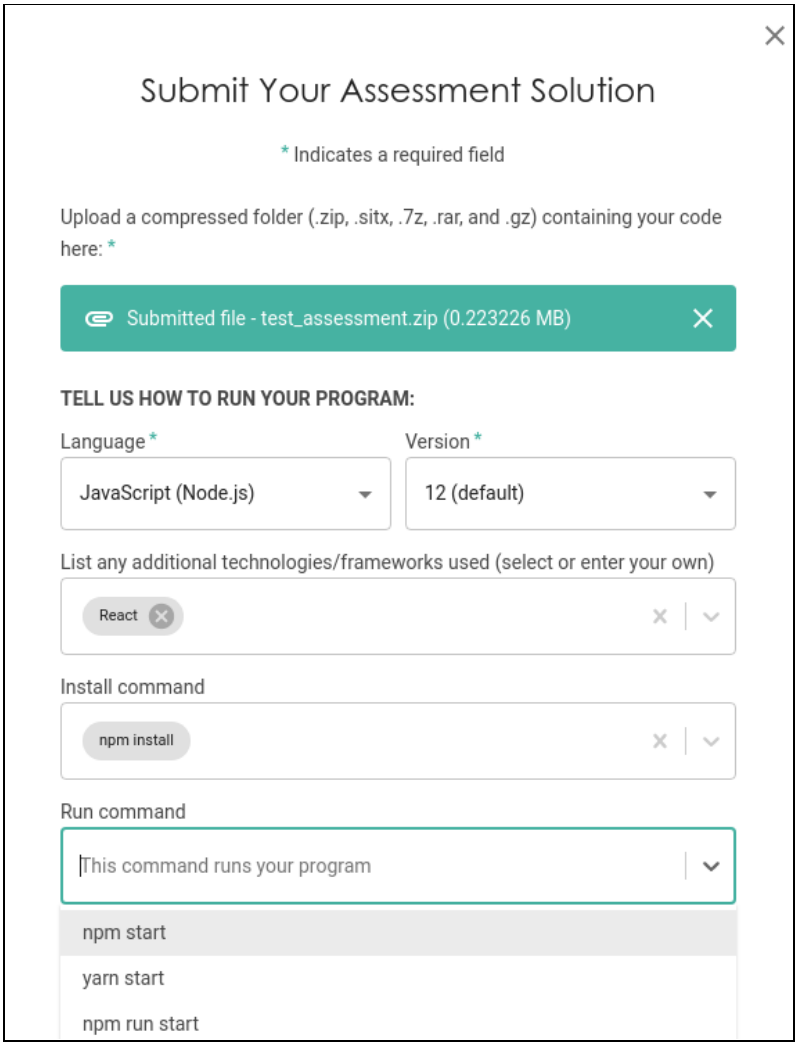
If you attempt to hardcode the paths or get them from the standard input, your application will not pass our tests.

Submission Details

Please submit your code in a compressed folder on the Hatchways platform. The max submission size is 5MB.

Upon clicking the submission button, you will see a form as pictured below. We need this information to be able to test your application.

1. Choose which language and technologies you used to develop your solution
2. Provide us with the **install commands**, and the **run command** that you used to run your application.
3. If you cannot find your commands in our suggestions, simply type your own and select "Use command".



Do not submit any built folders, since the compressed folder will be too large. **Do not submit your external dependencies (like the node_modules folder), since the compressed folder will be too large. We will be installing your dependencies before we run your code.**

If your submission is too big and you can't figure out how to compress, you are welcome to email your solution to hello@hatchways.io.

Please include your name, and use the email you signed up with on the Hatchways platform. Use the subject line “Software Assessment Submission”.

Public Repositories

Do not post your solution to a public repository. We understand that you may want to share projects you have worked on, but many hours go into developing our tools so we can provide a fair skills evaluation.