UNITED STATES
DIGITAL CORPS

Thank you for taking the time to participate in the project-based assessment phase of the U.S. Digital Corps hiring process. We are excited that you are moving forward with us!

All projects are due by 11:59pm ET on Tuesday, December 12, 2023.

Instructions: Please read through the below prompt thoroughly. Follow all directions and be sure to answer every question.

- You should plan to spend two hours completing this assessment.
- Please submit your code as a public git repository as described in the Submitting your work section. Submit your written responses in a PDF file of no more than three pages. Responses received in other formats may not be reviewed and written responses longer than three pages will only have the first three pages reviewed.
- Save your project using this naming convention:
    - Repository: "*firstnamelastname-usdc-2024*"
    - Written responses: "LastName_FirstName_Written_Software"
- **Email your completed assessment to usdigitalcorps+assessment@gsa.gov no later than 11:59pm ET on Tuesday, December 12, 2023.**
- **Do not share any details about this assessment with anyone, both during the assessment period and after the deadline has passed.**

Evaluation: Your submission will be reviewed by subject matter experts (SMEs) in your field and evaluated against the competencies and specialized experience shared in the job posting for this track. SMEs will look to understand your responses as well as your thought process.

Support: Please reach out to usdigitalcorps@gsa.gov with any questions or **to request an accommodation.**

This assessment is used solely for U.S. Digital Corps applicant evaluation.

UNITED STATES
DIGITAL CORPS

Project-Based Assessment – Software Engineering

Imagine that your organization has access to the scanned content of a number of books. You have been asked to write a function that searches for a word in the scanned book content.

Write a JavaScript function using the provided findSearchTermInBooks() signature. It should consume two inputs: a search term (a string) and a JSON object containing phrases from books. The function should return a single value (a JSON object) containing lines that match the search word. Example input and output objects are provided. Do not import modules for this exercise.

Input object

The input to the function will be a search term and a list of zero or more book objects. Each book object contains a list of zero or more pieces of scanned text.

The input object will have the following keys and types; all keys are in CamelCase. Example objects can be seen in the provided unit tests and below.

```
[
  {
    "Title": string,
    "ISBN": string,
    "Content": [
      {
        "Page": integer,
        "Line": integer,
        "Text": string
      }, zero or more...
    ],
  }, zero or more...
]
```

Note where we have indicated where there might be zero or more objects in the structure. A complete object might be a list containing no books, or it might be a list with just one book, but no scanned content!

The code in the repository has an example object that looks like the JSON structure below; it has one book object, but you can see that the book has three lines scanned, one of which has a hyphenated word break:

```
[
 {
   "Title": "Twenty Thousand Leagues Under the Sea",
   "ISBN": "9780000528531",
   "Content": [
    {
      "Page": 31,
      "Line": 8,
      "Text": "now simply went on by her own momentum.  The dark-"
    },
    {
    "Page": 31,
    "Line": 9,
    "Text": "ness was then profound; and however good the Canadian\'s"
    },
    {
    "Page": 31,
    "Line": 10,
    "Text": "eyes were, I asked myself how he had managed to see, and"
   }
  ]
 }
]
```

Output object

The output from the function will be a single JSON object. Contained within it will be the search term used and zero or more results from your search.

```
{
   "SearchTerm": string,
   "Results": [
     {
        "ISBN": string,
        "Page": integer,
        "Line": integer
     }, zero or more...
   ]
}
```

Again, you will always return a result object containing, at the least, the search term. The list of results could be empty.

The starter code has an example of output from a search for the word “the”. We include it here as well.

```
{
  "SearchTerm": "the",
  "Results": [
    {
      "ISBN": "9780000528531",
      "Page": 31,
      "Line": 9
    }
  ]
}
```

Only one line is found; note that the word “The” and the word “the” are *not* the same, because of capitalization.

Tasks

You have three tasks. You may complete them in whatever order you feel is appropriate.

1. **Implement the function** findSearchTermInBooks().
2. **Write unit tests** that demonstrate the correct execution of your implementation. At the least, you should write three kinds of tests:
   - Positive tests: tests that return a match.
   - Negative tests: tests that do not return any matches.
   - Case-sensitive tests: tests that match (for example) on “The” but not on “the”.

   There are many other ways to test your code; you are encouraged to think of other ways to test your code, and talk about them as part of your thought process document.

3. **Document your thought process**. This documentation should contain two parts. (You may include diagrams alongside written words, if you find them to be helpful.)

   - Overall process and decision making: The audience for this part of the document is your cross-functional team, which includes engineers, user researchers, and product managers. This means that everyone on the team has a technical background, but not everyone writes code. What they’re interested in is understanding how you arrived at your solution, so they can gain insight into your problem solving process. Avoid too much technical jargon, and use plain language as much as possible.

   - Testing and iteration: The audience for this part of the document is your team’s fellow engineers. Describe the following:

i. Your strategy for writing tests. If applicable, describe how, if you were given more time to work on this problem, you would seek to make the test suite more robust.

ii. What about your solution you are proudest of.

iii. Which part of the problem was most difficult to solve.

iv. Optional: Edge cases you addressed in your code, or how you might think about edge cases if you were given more time to work on this problem.

Starter template

The starter template is hosted in GitHub.

https://github.com/GSA-TTS/usdc-find-search-term-in-books

You can check this repository out on your local machine, and do your work in the file book_search.js. There are many tools you could use to check out your code; if you were working from a command line on a Mac Linux, or Windows machine, you might say:

```
git clone https://github.com/GSA-TTS/usdc-find-search-term-in-books
```

You can also download a .zip file containing that code directly from GitHub.

To run the code, you should open tester.html in your browser and look at the web developer console. The code should run as-is, and all the tests should (as provided) fail. The provided tests should pass once your code is complete and correct.

UNITED STATES
DIGITAL CORPS

Submitting your work

To submit your code, you will need to create a public git repository. This can be on GitHub, GitLab, Bitbucket, or any publicly accessible git repository.

You should name your repository: *firstnamelastname-usdc-2024*

For example, if you were a world-changing inventor, you might name your repository: *otisboykin-usdc-2024*

Because you are a future world-changing civic technologist, you should use your own first and last name.

The repository should contain two files: the tester.html file we provided to you and your solution in the file book_search.js. We only expect you to make edits to book_search.js.

To test that your repository is publicly accessible, try opening it from a browser in "Incognito mode." If you can access your code, then we should be able to as well.

Once you have your code in a repository that is publicly accessible, send us an email (usdigitalcorps+assessment@gsa.gov) that contains two things:

1. A link to your repository
2. Your thought process document as a PDF (LastName_FirstName_Written_Software)

Public Trust

We close with a note about public trust. You are applying for a position that asks you to hold the public's trust. The State Department has a page that talks about clearances, of which Public Trust is the entry point.

https://www.state.gov/security-clearances

You could search for solutions to this problem on the internet. You could ask someone to do the work for you. This is not a good way to start your career in civic tech. We encourage you to use documentation for JavaScript, articles on how to write good tests… in short, use resources to be successful. But we discourage you from trying to find ways to avoid demonstrating your own excellence by letting others do your work for you.