Technical Assessment Case Studies

The purpose of the Case Study is not only to gauge your technical ability, but also to see how you think. We will talk through your results here and your logic. The Case Study is not a make or break your candidacy for the role, but serve as a single data-point among your overall qualifications.

Please complete one of following case studies:

1. myRetail RESTful service
2. Barren Land Analysis
3. Search

For the case study you choose please meet the following requirements:

- Complete the exercise in the technical stack of your choice.
  - When appropriate use a data store of your choice.
  - Use any external frameworks you desire
  - Be ready to discuss your recommendations to make your solution suitable for use in a production environment

- Provide evidence of the result to the interviewers *(choose one)*
  - Unit test results or other documented output
  - Hosted instance of the implementation
  - Runnable instance of the implementation on your computer

- The end result should be a functional implementation of the problem with associated tests
  - Provide the working code either in a publicly accessible hosted repository or a zip file containing all necessary build steps and dependencies
  - Rename .js files to .js.txt if emailing code
  - Provide a README.md file with instructions for testing, running and interacting with your application and any details you feel are relevant to share

1. myRetail RESTful service

myRetail is a rapidly growing company with HQ in Richmond, VA and over 200 stores across the east coast. myRetail wants to make its internal data available to any number of client devices, from myRetail.com to native mobile apps.

The goal for this exercise is to create an end-to-end Proof-of-Concept for a products API, which will aggregate product data from multiple sources and return it as JSON to the caller.

Your goal is to create a RESTful service that can retrieve product and price details by ID. The URL structure is up to you to define, but try to follow some sort of logical convention.

Build an application that performs the following actions:

- Responds to an HTTP GET request at /products/{id} and delivers product data as JSON (where {id} will be a number.

Example product IDs: 13860428, 54456119, 13264003, 12954218)

- Example response: {"id":13860428,"name":"The Big Lebowski (Blu-ray) (Widescreen)","current_price":{"value": 13.49,"currency_code":"USD"}}
- Performs an HTTP GET to retrieve the product name from an external API. (For this exercise the data will come from redsky.target.com, but let's just pretend this is an internal resource hosted by myRetail)
- Example: http://redsky.target.com/v2/pdp/tcin/13860428?excludes=taxonomy,price,promotion,bulk_ship,rating_and_review_reviews,rating_and_review_statistics,question_answer_statistics
- Reads pricing information from a NoSQL data store and combines it with the product id and name from the HTTP request into a single response.
- BONUS: Accepts an HTTP PUT request at the same path (/products/{id}), containing a JSON request body similar to the GET response, and updates the product's price in the data store.

2. Barren Land Analysis

You have a farm of 400m by 600m where coordinates of the field are from (0, 0) to (399, 599). A portion of the farm is barren, and all the barren land is in the form of rectangles. Due to these rectangles of barren land, the remaining area of fertile land is in no particular shape. An area of fertile land is defined as the largest area of land that is not covered by any of the rectangles of barren land.

Read input from STDIN. Print output to STDOUT

Input

You are given a set of rectangles that contain the barren land. These rectangles are defined in a string, which consists of four integers separated by single spaces, with no additional spaces in the string. The first two integers are the coordinates of the bottom left corner in the given rectangle, and the last two integers are the coordinates of the top right corner.

Output

Output all the fertile land area in square meters, sorted from smallest area to greatest, separated by a space.

Sample Input	Sample Output
{“0 292 399 307”}	116800 116800
{“48 192 351 207”, “48 392 351 407”, “120 52 135 547”, “260 52 275 547”}	22816 192608

3. Document Search

The goal of this exercise is to create a working program to search a set of documents for the given search term or phrase (single token), and return results in order of relevance.

Relevancy is defined as number of times the exact term or phrase appears in the document.

Create three methods for searching the documents:

- Simple string matching
- Text search using regular expressions
- Preprocess the content and then search the index

Prompt the user to enter a search term and search method, execute the search, and return results. For instance:

> Enter the search term: <user enters search term>
>
> Search Method: 1) String Match 2) Regular Expression 3) Indexed
>
> Search results:
>
> File2.txt – X matches
>
> File1.txt - X matches
>
> File3.txt – X matches
>
> Elapsed time: 40 ms

Three files have been provided for you to read and use as sample search content.

Run a performance test that does 2M searches with random search terms, and measures execution time. Which approach is fastest? Why?

Provide some thoughts on what you would do on the software or hardware side to make this program scale to handle massive content and/or very large request volume (5000 requests/second or more).