Justia, Inc. Engineering Test

Instructions

If you are familiar with Github

- Create a branch based on `main` called `test`.
- For each of the following questions, you will need to commit your answer right after the "Answer" heading. It is fine to add images/attachments if needed as well. Make sure to commit your code to the `test` branch. **Important**: Please make sure that any additional attachment (code, image, etc.) is added inside `questions` folder, and not inside the root of the repository.
- When you are done, create a pull request from `test` branch to `main` branch.
- Inside the pull request should have any relevant notes you might think are necessary.
- Send an email with the link of the pull request to engineeringtest@justia.com.

If you are not familiar with Github

- Create a files with the answers for each question.
- Send an email to engineeringtest@justia.com with the answers as well as any other attachment that is relevant to the questions.

Notes

- If you have any questions, you can email the Justia Engineers (engineeringtest@justia.com), responses may take a few minutes so move on to the next question while you wait for a response.
- You have roughly 3 hours to complete the test, but you can email to request more time if needed.
- You can write the programming questions in any language you are familiar with.
- **Important:** Please review all the questions before you start with the test. The questions have different complexity. You can see a preview here (preview.pdf).

Questions

Question 1: Trademarks Database Modeling

Please show us how you would model the data to represent a “Trademark Document”. Below is an example of a trademark with many attributes related to it:

> https://trademarks.justia.com/884/84/justia-law-for-88484949.html (https://trademarks.justia.com/884/84/justia-law-for-88484949.html)

You should make sure to model the individual properties in the classification information and the Owners section. Please note that some things may have a 1:1 relationship, and others a 1:M relationship.

Your answer may be in SQL DDL statements, diagrams, or even as text so long as you effectively describe the schema of the database with its fields, types, and relationships.

Question 2: Split the line

Write a program in your favorite interpreted language to split a line of text into 2 lines as close to the same length as possible without splitting a word. If it is impossible to make the lines equal, the second line should be the longer one.

Example

Given:

> This is the first line and this is the second line, which is longer.

Expected result:

```
This is the first line and this
is the second line, which is longer.
```

Incorrect:

```
This is the first line and this is
the second line, which is longer.
```

```
This is the first line and this i
s the second line, which is longer.
```

```
This is the first line and this is the second line,
which is longer
```

```
This is the first line
and this is the second line, which is longer
```

Question 3: 8 Neighbors

Given a string of digits, find the largest number that you can get from the product of 8 neighboring digits. 8 neighboring digits would be 8 digits that are next to each other. In this example, 24568434 are the first set of neighboring digits and their product is 2*4*5*6*8*4*3*4 = 92,160. Write a program to do this below.

```
$digits =
‘24568434425600632669552439334106652406848309462638684828852262927278527936399
02871263148375668385161440704705082118433360826665181226306876475687530966949960
61243579876743129816446303352968183540538203615506141772107303222586640470832 39
20071844951365597184735205048431448632038265725533007537132986021894107034616 16
94702340482469490384492526133202783917955292141170629872596827195497292812036 15
42171054541369564051244167294448666871325462318813048565384739696273316952194 98
63912577861691287561977830934921212479165324543309497270273666798045961503058 39
92389560734400208254215719778167306362446897917842264846723630471017351940742 42
77432288002642013131483434176694348566477632943357495839257916341107854518712 14
88984723071987339436291800623771660042459548381224953144106574640745990955793 81
63127262469943694339330295089806219945292197456908348634614510839885078166623 4
23368121593055139117625532195547974797956352866878540104323878666136831467754 33
21962076490268925061893361115457644841490417310926106’
```

Question 4: 99 Beers

This is a "Code Golf" challenge. Code golf is a type of recreational computer programming competition in which participants strive to achieve the shortest possible source code that implements a certain algorithm:

- The shorter the better
- No restrictions on the programming language that you can use
- Don't worry about optimizations, readability, and maintainability.
- The language knowledge & mastery is preferred over the algorithmic skills
- Don't copy & print directly from the source file

The challenge is to write a script that reproduces the lyrics of the song “99 Bottles of Beer” and send them to standard output. You can find the lyrics here (question04-file.txt).

Your script needs to:

- **Output the exact same content as the text file above.**
- Be as short in character length as possible.

Question 5: State Codes

Let's suppose we have the following states array:

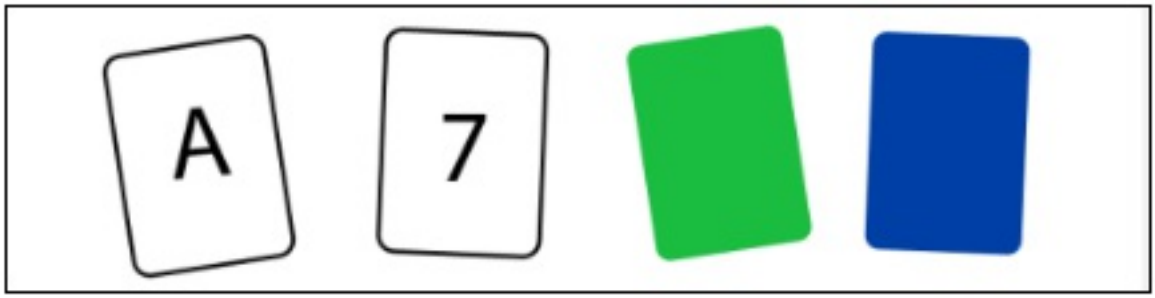
```
[
    'al' => 'alabama',
    'ak' => 'alaska',
    'az' => 'arizona',
    'ar' => 'arkansas',
    'ca' => 'california',
    'co' => 'colorado',
    'ct' => 'connecticut',
    'de' => 'delaware',
    'dc' => 'district of columbia',
    'fl' => 'florida',
    'ga' => 'georgia',
    'hi' => 'hawaii',
```

```
    'id' => 'idaho',
    'il' => 'illinois',
    'in' => 'indiana',
    'ia' => 'iowa',
    'ks' => 'kansas',
    'ky' => 'kentucky',
    'la' => 'louisiana',
    'me' => 'maine',
    'md' => 'maryland',
    'ma' => 'massachusetts',
    'mi' => 'michigan',
    'mn' => 'minnesota',
    'ms' => 'mississippi',
    'mo' => 'missouri',
    'mt' => 'montana',
    'ne' => 'nebraska',
    'nv' => 'nevada',
    'nh' => 'new hampshire',
    'nj' => 'new jersey',
    'nm' => 'new mexico',
    'ny' => 'new york',
    'nc' => 'north carolina',
    'nd' => 'north dakota',
    'oh' => 'ohio',
    'ok' => 'oklahoma',
    'or' => 'oregon',
    'pa' => 'pennsylvania',
    'ri' => 'rhode island',
    'sc' => 'south carolina',
    'sd' => 'south dakota',
    'tn' => 'tennessee',
    'tx' => 'texas',
    'ut' => 'utah',
    'vt' => 'vermont',
    'wa' => 'washington',
    'wv' => 'west virginia',
    'va' => 'virginia',
    'wi' => 'wisconsin',
    'wy' => 'wyoming',
    'gu' => 'guam',
    'mp' => 'mariana islands',
    'pr' => 'puerto rico',
    'ci' => 'circuit courts',
    'vi' => 'virgin islands'
]
```

Let's print them each one, on a separate line, alpha-sorted by the state code, and with the following format:

```
T.X. is the code for Texas
```

Question 6: Pick the right cards



Having the following statement:

> If a number is on one side, the opposite side is green.

From the 4 cards above, which cards do you absolutely need to turn in order to validate that the statement above is true?

Question 7: Big O notation

Do you know what is the Big O notation for this function?

```
function pairMultiplications(numbers) {
    console.log('these are the numbers: ')
    numbers.forEach(function (number) {
        console.log(number);
    });

    console.log('and these are their multiplications: ')
    numbers.forEach(function (firstNumber) {
        numbers.forEach(function (secondNumber) {
            console.log(firstNumber * secondNumber);
        });
    });
}

pairMultiplications([1,2,3,4,5]);
```

Question 8: The Best in Her Class

Imagine that there's a girl that is the best in her class of mathematical statistics and probability.

What's more likely to happen, and explain why:

- She goes on to become a painter
- She goes on to become a painter that also plays poker in her free time

Question 9: Building a Car

When putting together a wheel for a car, one picks a tire that is compatible with a rim.

```
<?php

class Wheel
{
    private $tire;
    private $rim;

    public function __construct ($width, $diameter, $section_width,
$aspect_ratio)
    {
        $this->tire = new Tire($section_width, $diameter, $aspect_ratio);
        $this->rim = new Rim($width, $diameter);
    }

    public function getTire()
    {
        return $this->tire;
    }

    public function getRim()
    {
        return $this->rim;
    }
}

class Tire
{
    private $section_width;
    private $diameter;
    private $aspect_ratio;

    public function __construct($section_width, $diameter, $aspect_ratio) {/*
... */}

    // Getters & Setters
}

class Rim
{
        private $rim_width;
        private $diameter;

        public function __construct($rim_width, $diameter) {/* ... */}

        // Getters & Setters
}
```

The above code violates the dependency inversion principle which causes some tight coupling of classes. Use dependency injection to recode the Wheel, Tire, and Rim classes. Please rewrite this in any language.

Question 10: Pagination

Create a pagination class that takes the following constructor arguments:

- An array of items. (No specific type, these are the items that we are paginating.)
- Items per page argument.
- Current page.

It must also define the following public methods:

- `nextPage()` // Returns null or the integer of the next page.
- `previousPage()` // Returns null or the integer of the previous page.
- `currentItems()` // Returns an array of the current items to show in the page.
- `itemsOnPage($n)` // Returns the array of items for the n-th page.
- `firstPage()` // Returns the integer of the first page.
- `lastPage()` // Returns the integer of the last page.

Question 11: Inheritance vs traits

Please list as many technical differences as you can between Inheritance and Traits in PHP. No code is necessary, a bullet point list will do.