onebitcode{}
FOR AMAZING PROGRAMMERS

30 DESAFIOS de JavaScript

- Com resolução **do básico ao expert**

{}

SUA JORNADA

Você acaba de ser recrutado(a)

para uma jornada na galáxia com o objetivo
de cumprir 30 desafios de JavaScript
do nível Very Easy ao Very Hard
e conquistar 5 atribuições!

onebitcode
FOR AMAZING PROGRAMMERS

Tire um print, publique nos stories do Instagram
e marque @onebitcode

SUMÁRIO

Módulo Very Easy

1. Ajudando a Academia.....02
2. Procedimento Recursivo I.....04
3. Linguagem Limitada.....06
4. Cálculos de Viagens Espaciais I.....08
5. História Escondida I.....10
6. História Escondida II.....12

Módulo Easy

7. Organizando Resultados.....16
8. Biblioteca Interplanetária.....18
9. Instruções de Emergência.....20
10. Procedimento Recursivo II.....22
11. Intervalo de Coordenadas.....24
12. Espaçoporto Comercial.....26
13. Código de Identificação de Nave.....28
14. Licença de Voo.....31

Módulo Medium

15. Código Romano I.....36
16. Código Romano II.....39
17. Relatório de Missão.....41
18. Módulo de Treinamento.....43
19. Cálculos de Viagens Espaciais II.....46
20. Quebrando a Senha.....47
21. Sistema de Localização.....49
22. Prisão Intergalática.....52

Módulo Hard

23. Corrida de Pods.....57
24. Validações de Usuário.....60
25. Validações de Compilador.....63
26. Otimização de Cálculos.....65
27. Encontrando Conexões.....67
28. Hora de Jogar.....71

Módulo Very Hard

29. Labirintos Subterrâneos.....81
30. Planilha Portátil.....85

nível

VERY EASY

Arrays Matemática

1. Ajudando a Academia

A Academia de Oficiais da Federação precisa da sua ajuda para criar um programa que automatize os cálculos das médias de alunos e turmas para melhor acompanhar os seus desempenhos.

Este programa será incorporado em seu sistema principal, portanto você precisa apenas desenvolver a função que calcula a média.

No entanto, a função deve ser capaz de funcionar com qualquer número de alunos ou turmas.

Desafio 1: Ajudando a Academia

Escreva uma função que recebe um número qualquer de números inteiros como argumentos e retorne a média aritmética entre eles.

Testes 1: Ajudando a Academia

- Entrada: (10, 9, 6, 8, 9, 1, 5, 7)
Saída: 6.875

- Entrada: (2, 5, 7, 1, -2)
Saída: 2.6

- Entrada: (10, 10, 10, 10, 9)
Saída: 9.8

- Entrada: (25, 75)
Saída: 50

Recursão Strings

2. Procedimento Recursivo I

A equipe de manutenção da nave Highwind solicitou a sua ajuda para analisar o desempenho do computador portátil utilizado pela equipe de reconhecimento.

Para isso, em um determinado momento, você precisou simular a criação de blocos de informação em formato de texto recursivamente, então decidiu criar uma função para imitar esse comportamento.

Desafio 2: Procedimento Recursivo I

Escreva uma função que recebe um número e retorna uma quantidade equivalente de "chunks" separados por um traço "-" sem utilizar nenhuma estrutura de repetição (while, dowhile, for, etc).

Testes 2: Procedimento Recursivo I

- Entrada: (4)
  Saída: "chunk-chunk-chunk-chunk"

- Entrada: (1)
  Saída: "chunk"

- Entrada: (8)
  Saída:
"chunk-chunk-chunk-chunk-chunk-chunk-chunk-chunk"

- Entrada: (2)
  Saída: "chunk-chunk"

Arrays Estruturas Básicas

3. Linguagem Limitada

Um dispositivo portátil encontrado no planeta Anelius está sendo estudado pela equipe de pesquisadores do Instituto de Tecnologia da Federação.

Ele funciona a base de uma linguagem desconhecida e limitada.

Para estender suas funcionalidades, foi solicitado que você crie um procedimento que seja capaz de inverter listas de forma não destrutiva, ou seja, sem alterar a lista original, porém utilizando apenas recursos básicos de repetição (for, while, etc).

Desafio 3: Linguagem Limitada

Escreva uma função que recebe um array e retorne um novo array com todas as posições invertidas do original sem alterá-lo. Não utilize os métodos do objeto global Array do javascript (reverse, map, forEach, etc).

Testes 3: Linguagem Limitada

- Entrada: [0, 9, 6, 8, 9, 1, 5, 7]
Saída: [7, 5, 1, 9, 8, 6, 9, 0]

- Entrada: ['Oh', 'Hi', 'Mark']
Saída: ['Mark', 'Hi', 'Oh']

- Entrada: [false, true, true, true]
Saída: [true, true, true, false]

- Entrada: ["It's", "not", true, 0]
Saída: [0, true, 'not', "It's"]

Matemática Manipulação de Tipos

4. Cálculos de Viagens Espaciais I

A equipe encarregada de aperfeiçoar o sistema de navegação das naves da Federação precisa que você os ajude criando uma nova funcionalidade.

Essa funcionalidade consiste em calcular a elevação quadrática individual de um determinado número.

Para realizar esse cálculo você deve elevar ao quadrado cada algarismo do número em questão e concatenar os resultados em um único número.

Desafio 4: Cálculos de Viagens Espaciais I

Escreva uma função que recebe um número inteiro qualquer, eleve ao quadrado cada um de seus algarismos e depois concatene o resultado retornando um único número inteiro.

Testes 4: Cálculos de Viagens Espaciais I

- Entrada: (3514)
Saída: 925116

- Entrada: (94571)
Saída: 811625494

- Entrada: (24)
Saída: 416

- Entrada: (745821698)
Saída: 4916256441368164

Strings Arrays

5. História Escondida I

Durante uma missão de exploração no planeta Ondur a tropa de reconhecimento encontrou tabuletas contendo escritos de uma antiga civilização, mas ao traduzir seu conteúdo as mensagens não faziam sentido.

Porém, um pesquisador que estudava os escritos percebeu que poderiam haver mensagens escondidas.

Como os escritos eram muitos e para poder analisar melhor ele solicitou que você criasse um programa que o ajudasse a encontrar a história escondida contada pelas tabuletas.

Para conseguir isso o programa precisar ser capaz de identificar a maior letra (segundo a ordem alfabética) de cada trecho dos escritos.

Desafio 5: História Escondida I

Escreva uma função que recebe uma string e retorna a maior letra segundo a ordem alfabética em minúsculo. Assuma que a string não possui nenhuma letra com acento, número ou caractere especial, apenas letras e espaços.

Testes 5: História Escondida I

- Entrada: ('Lorem ipsum dolore sec avanti')
Saída: 'v'

- Entrada: ('Hello')
Saída: 'o'

- Entrada: ('May the force be with you')
Saída: 'y'

- Entrada: ('Its over nine thousand')
Saída: 'v'

Strings Arrays

6. História Escondida II

Ainda analisando as tabuletas do planeta Ondur, dessa vez o pesquisador encontrou um padrão diferente nos escritos.

Ele agora precisa que você atualize o programa para ser capaz de ler esse novo padrão.

Para isso, o programa precisa ser capaz de inverter as palavras dos trechos dos escritos, porém sem alterar a ordem em que elas estão escritas.

Desafio 6: História Escondida II

Escreva uma função que recebe uma string e retorna cada palavra da string invertida e em letras minúsculas, porém com as palavras na mesma ordem.

Assuma que a string não possui nenhuma letra com acento, número ou caractere especial, apenas letras e espaços.

Testes 6: História Escondida II

- Entrada: ('Lorem ipsum dolore sec avanti')
Saída: 'merol muspi erolod ces itnava'

- Entrada: ('This is an apple')
Saída: 'siht si na elppa'

- Entrada: ('May the force be with you')
Saída: 'yam eht ecrof eb htiw uoy'

- Entrada: ('It s over nine thousand')
Saída: 'ti s revo enin dnasuoht'

PARABÉNS!

Você acaba de concluir
6 desafios Very Easy e avançar
na sua jornada pela galáxia.

Sua nova atribuição:

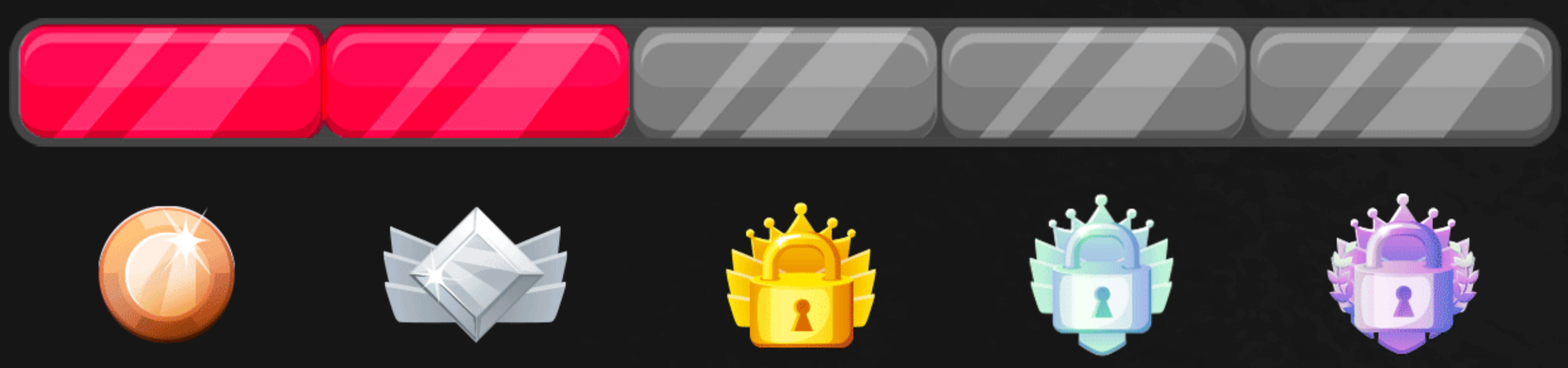
Programador(a) JavaScript
das galáxias nível bronze.

onebitcode{}
FOR AMAZING PROGRAMMERS

Tire um print, publique nos stories do Instagram
e marque @onebitcode

nível

EASY



Arrays

7. Organizando Resultados

O sistema interno da nave realiza periodicamente uma verificação de seus principais componentes para checar sua integridade.

Os vários resultados dessa verificação são emitidos em formato de listas de referências numéricas desorganizadas.

Para otimizar isso você deve criar um procedimento que seja capaz de organizar todas as listas de números em uma única lista ordenada.

Desafio 7: Organizando Resultados

Escreva uma função que recebe um array bidimensional de inteiros e retorna um único array contendo todos os números em ordem ascendente.

Testes 7: Organizando Resultados

- Entrada: ([[1, 5, 3], [6, 19, 11], [47, 128, 5], [1, 93, 57, 42, 103]])
Saída: [1, 1, 3, 5, 5, 6, 11, 19, 42, 47, 57, 93, 103, 128]

- Entrada: ([[1, 3], [4, 8], [7, 5], [2, 6]])
Saída: [1, 2, 3, 4, 5, 6, 7, 8]

- Entrada: [[], [], [], []]
Saída: [ ]

- Entrada: ([[100, 50], [60, 100], [20, 100, 70], [10, 40, 80, 90]])
Saída: [10, 20, 40, 50, 60, 70, 80, 90, 100, 100, 100]

Strings

8. Biblioteca Interplanetária

A Biblioteca Interplanetária acumula conhecimento de todas as partes da galáxia, possuindo um acervo gigantesco e muito rico.

Todas as publicações são organizadas por autor e para isso o último nome do autor é utilizado para identificação.

A direção da biblioteca precisa que você crie um programa para converter o nome completo de um autor para um nome abreviado corretamente no formato utilizado na organização das publicações.

Desafio 8: Biblioteca Interplanetária

Escreva uma função que recebe uma string contendo um nome completo e retorna uma string com todos os nomes, exceto o último, abreviados e o último nome em letras maiúsculas antes das abreviações separado por vírgula.

Testes 8: Biblioteca Interplanetária

- Entrada: ('Isaac Larrubia Ferreira Pontes')
Saída: 'PONTES, I. L. F.'

- Entrada: ('John Ronald Reuel Tolkien')
Saída: 'TOLKIEN, J. R. R.'

- Entrada: ('christopher james paolini')
Saída: 'PAOLINI, C. J.'

- Entrada: ('Suzanne Marie Collins')
Saída: 'COLLINS, S. M.'

Strings

9. Instruções de Emergência

As estações e naves de maior porte da Frota Estelar em situações de emergência precisam enviar instruções de navegação importantes e que não podem ser comprometidas.

Para isso, algumas validações precisam ser feitas nos códigos de transmissão.

Uma dessas validações envolve um código que deve possuir a mesma quantidade de cada caractere que o compõe. Foi solicitado a você que crie a função que validará este código.

Desafio 9: Instruções de Emergência

Escreva uma função que recebe uma string, verifica se ela possui a mesma quantidade de cada letra que a compõe e retorna true caso possua ou false caso não possua.

Testes 9: Instruções de Emergência

- Entrada: ('This is Thee')
Saída: true

- Entrada: ('ssd')
Saída: false

- Entrada: ('Lorem ipsum')
Saída: false

- Entrada: ('QQwweerrttyy')
Saída: true

Recursão Matemática

10. Procedimento Recursivo II

Mais uma vez a equipe de manutenção solicitou a sua ajuda para analisar o desempenho dos computadores portáteis da equipe de reconhecimento.

Dessa vez, no entanto, o procedimento que você ficou encarregado de testar envolve cálculos matemáticos, e você irá utilizar o cálculo do fatorial de um determinado número para isso.

Portanto, será preciso criar um procedimento que seja capaz de calcular o fatorial de qualquer número inteiro positivo sem utilizar estruturas de repetição.

Desafio 10: Procedimento Recrusivo II

Escreva uma função que recebe um número e retorna o seu fatorial sem utilizar nenhuma estrutura de repetição (while, dowhile, for, etc).

O fatorial de um númeroé igual a multiplicação de todos os inteiros positivos menores ou iguais a ele. Ela deve ser capaz de retornar números inteiros corretos mesmo para valores altos.

Testes 10: Procedimento Recursivo II

- Entrada: (5)
Saída: 120n

- Entrada: (0)
Saída: 1n

- Entrada: (32)
Saída: 263130836933693530167218012160000000n

- Entrada: (9n)
Saída: 362880n

Arrays

11. Intervalo de Coordenadas

O capitão da sua nave precisa que você implemente no sistema principal uma forma de obter todas as coordenadas abaixo de um determinado ponto partindo da origem (0, 0) e retorne isso para o sistema em uma lista ordenada crescente. As coordenadas são escritas na forma de pares ordenados (x, y).

Desafio 11: Intervalo de Coordenadas

Escreva uma função que receba um par ordenado (x, y) e retorne um array de pares (x, y) onde cada um deles possui x e y menor ou igual ao par recebido em ordem crescente.

Os pares devem ser ordenados de forma que primeiro aumente o valor de y e depois o valor de x.

Apenas o quadrante onde x e y são positivos deve ser considerado.

Testes 11: Intervalo de Coordenadas

- Entrada: ([2, 2])
Saída: [[0, 0], [0, 1], [0, 2], [1, 0], [1, 1], [1, 2], [2, 0], [2, 1], [2, 2]]

- Entrada: ([2, 7])
Saída: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [0, 7], [1, 0], [1, 1], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [1, 7], [2, 0], [2, 1], [2, 2], [2, 3], [2, 4], [2, 5], [2, 6], [2, 7]]

- Entrada: ([-3, -3])
Saída: [ ]

- Entrada: ([7, 6])
Saída: [[0, 0], [0, 1], [0, 2], [0, 3], [0, 4], [0, 5], [0, 6], [1, 0], [1, 1], [1, 2], [1, 3], [1, 4], [1, 5], [1, 6], [2, 0], [2, 1], [2, 2], [2, 3], [2, 4], [2, 5], [2, 6], [3, 0], [3, 1], [3, 2], [3, 3], [3, 4], [3, 5], [3, 6], [4, 0], [4, 1], [4, 2], [4, 3], [4, 4], [4, 5], [4, 6], [5, 0], [5, 1], [5, 2], [5, 3], [5, 4], [5, 5], [5, 6], [6, 0], [6, 1], [6, 2], [6, 3], [6, 4], [6, 5], [6, 6], [7, 0], [7, 1], [7, 2], [7, 3], [7, 4], [7, 5], [7, 6]]

Matemática

12. Espaçoporto Comercial

A sua nave está de passagem por um planeta que usa como dinheiro as moedas de créditos.

Existem moedas de 1, 5, 10, 25, 100 e 500 créditos.

Para conseguir um favor de um comerciante local você irá ajudá-lo a automatizar o processo de contagem das moedas através de um programa que calcula quantas de cada moeda ele irá precisar para chegar a um valor determinado.

Desafio 12: Espaçoporto Comercial

Escreva uma função que receba um valor inteiro e retorne a quantidade de cada moeda para se chegar ao valor.

Deve-se sempre priorizar as moedas de maior valor (o máximo possível de moedas de 500, depois o máximo possível de moedas de 100, etc).

Testes 12: Espaçoporto Comercial

- Entrada: (478)
Saída: { '1': 3, '5': 0, '10': 0, '25': 3, '100': 4, '500': 0 }

- Entrada: (384)
Saída: { '1': 4, '5': 1, '10': 0, '25': 3, '100': 3, '500': 0 }

- Entrada: (5412)
Saída: { '1': 2, '5': 0, '10': 1, '25': 0, '100': 4, '500': 10 }

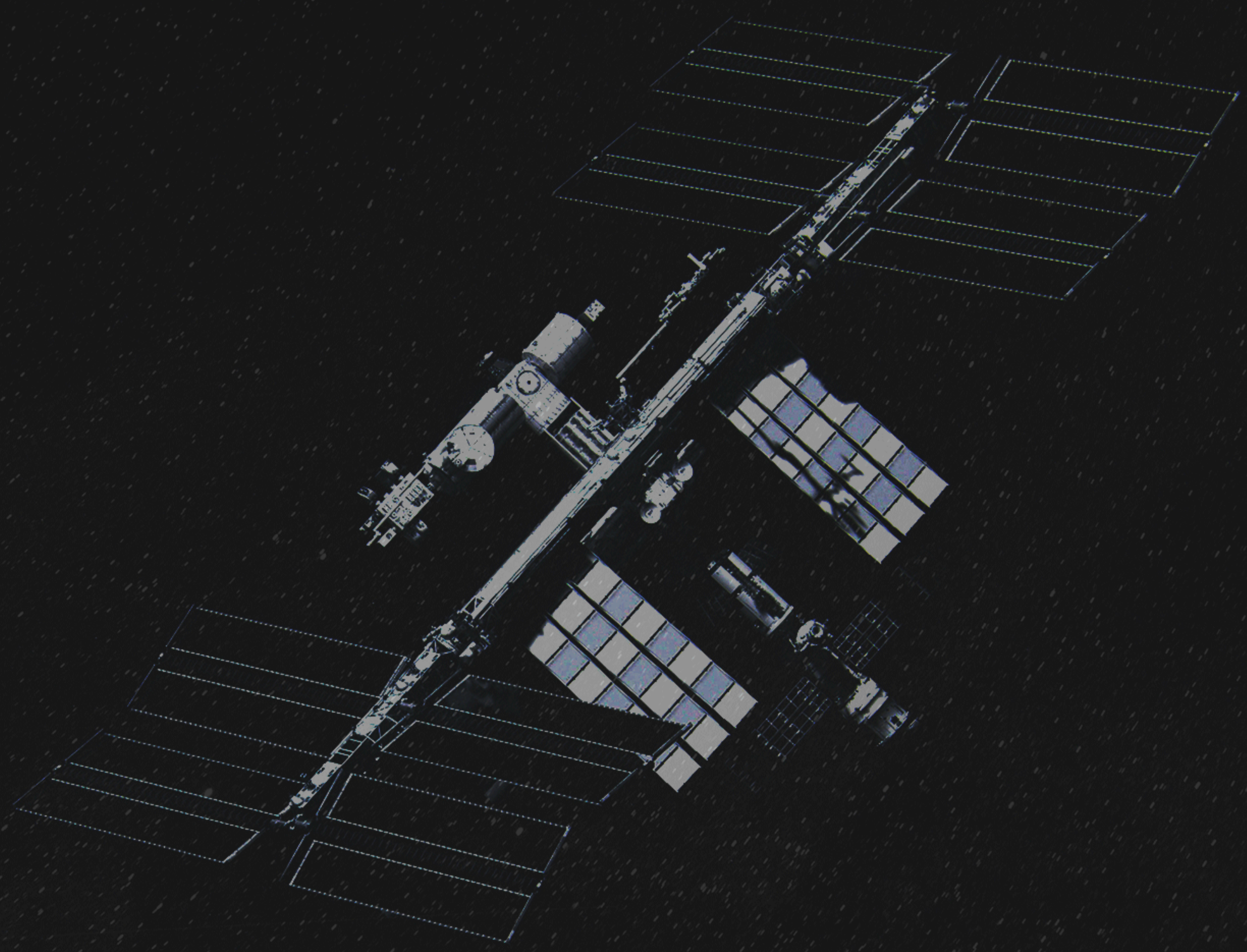
- Entrada: (50)
Saída: { '1': 0, '5': 0, '10': 0, '25': 2, '100': 0, '500': 0 }

Strings Matemática

13. Código de Identificação de Nave

As naves da Federação utilizam um código único que serve para identificá-las. Esse código é composto de 12 algarismos, sendo o último um dígito verificador.

Você está construindo em sua própria nave um módulo capaz de analisar um código de identificação recebido e verificar se é um código válido de uma nave da Federação.



Desafio 13: Código de Identificação de Nave

Escreva uma função que recebe um número e verifica se ele é um código de identificação válido segundo as regras de criação do dígito verificador. A criação do dígito verificador funciona da seguinte forma:

1. Some os dígitos das posições pares (ímpares se estiver contando a partir de 1)

2. Multiplique esse resultado por 3

3. Some os dígitos das posições ímpares (pares se estiver contando a partir de 1) do número original e então some esse resultado ao resultado do passo anterior

4. Encontre o resto da divisão do resultado do passo anterior por 10

5. Se o resto da divisão for 0, o dígito verificador é 0, do contrário o dígito verificador é 10 - resto

Testes 13: Código de Identificação de Nave

- Entrada: (5470207437 89)
Saída: true

- Entrada: (301354030348)
Saída: true

- Entrada: (301354030349)
Saída: false

- Entrada: (123456789872)
Saída: false

Strings Classes Datas

14. Licença de Voo

Para pilotar uma nave de pequeno porte em qualquer planeta da Federação é preciso possuir uma licença de voo.

O código usado na licença de voo é criado a partir de informações de seu dono, como nome e data de nascimento.

Foi solicitado que você crie um método para gerar a licença de voo de uma determinada pessoa para ser integrado aos sistemas da Federação. É obrigatório fazer isso utilizando uma classe.

Desafio 14: Licença de Voo

Escreva uma classe que contenha um método para gerar uma licença de voo e os seguintes atributos:

- Nome
- Sobrenome
- Data de Nascimento
- Licença de Voo (que deve iniciar sempre como falso)

Além disso a classe deve possuir um método para criar uma licença caso a pessoa ainda não possua uma. A licença deve ser uma string seguindo o seguinte padrão:

- Os primeiros cinco caracteres do sobrenome em letras maiúsculas (completado com 9's caso possua menos de cinco)

- O 6º caractere é um traço (-)

- O 7º caractere é o algarismo da década (penúltimo) do ano de nascimento

- O 8º e 9º caracteres são o mês de nascimento

- O 10º caractere é o algarismo do ano (último) do ano de nascimento

- O 11º caractere é um ponto (.)

- O 12º caractere é a primeira letra do primeiro nome (minúscula)

Testes 14: Licença de Voo

```
- Pilot {
  firstName: 'John',
  lastName: 'Doe',
  birthday: 1977-05-25T03:00:00.000Z,
  flyingLicense: 'DOE99-7057.j'
}
- Pilot {
  firstName: 'Hal',
  lastName: 'Jordan',
  birthday: 1995-09-02T03:00:00.000Z,
  flyingLicense: 'JORDA-9095.h'
}
Pilot {
  firstName: 'Carol',
  lastName: 'Danvers',
  birthday: 1968-08-17T03:00:00.000Z,
  flyingLicense: 'DANVE-6088.c'
}
Pilot {
  firstName: 'Poe',
  lastName: 'Dameron',
  birthday: 1979-03-09T03:00:00.000Z,
  flyingLicense: 'DAMER-7039.p'
}
```

PARABÉNS!

Você acaba de concluir
8 desafios Easy e avançar
na sua jornada pela galáxia.

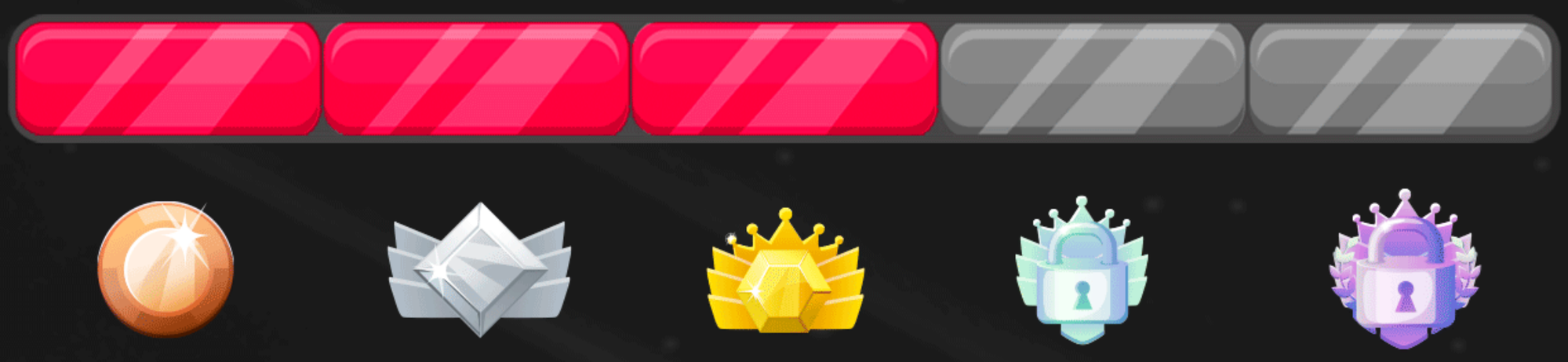
Sua nova atribuição:
Programador(a) JavaScript
das galáxias nível prata.

onebitcode
FOR AMAZING PROGRAMMERS

Tire um print, publique nos stories do Instagram
e marque @onebitcode

nível

MEDIUM



Matemática | Manipulação de Tipos

15. Código Romano I

Em uma missão por regiões não mapeadas pela Federação sua equipe encontrou um povoado que utilizava um sistema numérico muito parecido com o romano.

Você foi encarregado de implementar um tradutor capaz de converter qualquer número (em formato de texto) para o seu inteiro decimal correspondente.

Desafio 15: Código Romano I

Escreva uma função que recebe uma string de algarismos romanos e seja capaz de traduzir seu conteúdo retornando o inteiro decimal correspondente.

Os algarismos romanos são:

- I: 1
- V: 5
- X: 10
- L: 50
- C : 100
- D : 500
- M: 1000

Os outros números são formados a partir de dois tipos de notação, a padrão e a subtrativa.

- Notação Padrão: I, II e III (1, 2 e 3), VI, VII e VIII (6, 7 e 8), X, XX e XXX (10, 20 e 30), etc.

- Notação Subtrativa: IV (1 - 5 = 4), IX (1 - 10 = 9), XLIX (10 - 50 + 1 - 10 = 49), XC (10 - 100 =90), CMXCIX (100 - 1000 + 10 - 100 + 1 - 10 = 999)

Testes 15: Código Romano I

- Entrada: ('XLVII')
Saída: 47

- Entrada: ('CDXXXVIII')
Saída: 438

- Entrada: ('CMIX')
Saída: 909

- Entrada: ('MMMCMXCIX')
Saída: 3999

Matemática | Manipulação de Tipos | Strings

16. Código Romano II

Enquanto em contato com o mesmo povoado do planeta não mapeado, você e sua equipe interceptaram uma tentativa de comunicação que estava sendo enviada para algum lugar.

O conteúdo parecia estar criptografado utilizando um método primitivo semelhante à cifra de Cesar.

Agora vocês precisam de um programa capaz de traduzir esse conteúdo a partir de uma chave.

Desafio 16: Código Romano II

Escreva uma função que recebe uma string e um número e retorne o resultado da aplicação da Cifra de Cesar para descriptografar o seu conteúdo, ou seja, que retroceda cada letra pelo número passado seguindo a ordem alfabética.

Testes 16: Código Romano II

- Entrada: ('Vguvg', 2)
Saída: 'Teste'

- Entrada: ('BCDYZAbcdyza', 27)
Saída: 'ABCXYZabcxyz'

- Entrada: ('Qaiik', 60)
Saída: 'Isaac'

- Entrada: ('Amozmlw', 8)
Saída: 'Segredo'

Strings Arrays

17. Relatório de Missão

Ao concluir uma missão em nome da Federação o capitão do esquadrão deve redigir o seu relatório como parte do protocolo padrão.

Para auxiliar neste processo, um robô assistente revisa o texto e pode sugerir correções.

A fim de agilizar esse processo, foi requisitado que você ajudasse na construção de uma interface para que o robô insira as correções diretamente no arquivo do relatório.

Desafio 17: Relatório de Missão

Escreva uma função que recebe três argumentos, uma frase, uma palavra e um array de índices. A função deve retornar a frase com a palavra inserida em cada uma das posições especificadas pelo array de índices.

- A função não deve funcionar em índices que não estejam no alcance da frase.
- A função deve retornar a mesma frase caso o array de índices esteja vazio.

Testes 17: Relatório de Missão

- Entrada: ('capaz utilizar as cápsulas emergência', 'de ', [6, 27])
Saída: 'capaz de utilizar as cápsulas de emergência'

- Entrada: ('Nós decidimos apesar das chances serem baixas que enviaríamos um sinal para [...]', ',', [13, 45])
Saída: 'Nós decidimos, apesar das chances serem baixas, que enviaríamos um sinal para [...]'

- Entrada: ('Hello', 'world', [6])
Saída: 'Hello'

- Entrada: ('Isso é um teste', 'não', [])
Saída: 'Isso é um teste'

Classes Matemática

18. Módulo de Treinamento

Na colônia estabelecida no planeta primitivo Romulus está sendo criada uma escola a fim de fornecer educação básica para os colonos.

Para auxiliar durante as aulas de matemática, foi solicitado que uma equipe construísse um programa de computador capaz de explicar e resolver diversos tipos de problemas e conceitos.

Como parte da equipe, você foi encarregado de construir o módulo que trata de equações até o segundo grau, portanto precisará construir um programa capaz de resolver equações e descrever os passos da resolução.

É obrigatório o uso de classes.

Desafio 18: Módulo de Treinamento:

Escreva uma classe Equation que seja capaz de lidar com uma equações até o segundo grau.

Os objetos podem ser instanciados com valores padrões para a = 0, b = 0 e c = 0.

Ela deve ter um método para retornar suas raízes na forma de um array vazio, de um ou de dois elementos.

O método também deve descrever os passos para a resolução da equação.

Se a, b e c forem todos iguais a 0 ela não deve calcular as raízes e deve retornar uma mensagem de erro.

Testes 18: Módulo de Treinamento

- Entrada: (0, 2, 6)
Saída:
  1. 2x + 6 = 0
  2. 2x = -6
  3. x = -6 / 2
  4. x = -3
  [-3]

- Entrada: ()
Saída: 'Erro! Nenhum parâmetro informado.'

- Entrada: (0, 0, 5)
Saída:
  1. Parâmetros insuficientes.
  Nenhuma raiz real.
  [ ]

- Entrada: (1, -4, -5)
Saída:
  1. Δ = -4² - 4 * 1 * -5
  2. Δ = 36
  3. x' = (-(-4) + √36) / 2 * 1
  4. x" = (-(-4) - √36) / 2 * 1
  5. x' = 4 + √36 / 2
  6. x" = 4 - √36 / 2
  7. x' = 4 + 6 / 2
  8. x" = 4 - 6 / 2
  9. x' = 5
  10. x" = -1

  [5, -1]

Matemática Recursão

19. Cálculos de Viagens Espaciais II

A equipe de desenvolvimento do sistema de navegação das naves da Federação precisa novamente da sua ajuda para implementar uma funcionalidade.

Ela consiste em calcular a persistência multiplicativa de um determinado número.

A persistência multiplicativa é a quantidade de vezes que se precisa substituir um número pelo produto de seus algarismos até chegar a um número de um único dígito.

Por exemplo: 539 = 3, pois 5 x 3 x 9 = 135, 1 x 3 x 5 = 15 e 1 x 5 = 5, ou seja, 3 multiplicações.

Desafio 19: Cálculos de Viagens Espaciais II

Escreva uma função que recebe um número e retorna a sua persistência multiplicativa, que é a quantidade de vezes que é necessário multiplicar os seus algarismos para se chegar em um número de um único algarismo.

Testes 19: Cálculos de Viagens Espaciais II

- Entrada: (539)
Saída: 3

- Entrada: (999)
Saída: 4

- Entrada: (7)
Saída: 0

- Entrada: (7169)
Saída: 5

Recursão Arrays

20. Quebrando a Senha

Em uma missão no planeta Darnas a equipe de reconhecimento encontrou diversos dispositivos que parecem ter informações cruciais para a missão, porém os dispositivos estão bloqueados por senhas.

Você e seu colega estão criando um procedimento para descobrir as senhas utilizando força bruta.

Você ficou encarregado de criar a função que irá calcular todas as senhas possíveis a partir da lista de caracteres que a compõe.

Desafio 20: Quebrando a Senha

Escreva uma função que receba um array de opções e retorne um array bidimensional de todas as senhas possíveis utilizando todos os elementos passados. Faça isso utilizando recursão.

Testes 20: Quebrando a Senha

- Entrada: (["X", "s", "-", "#"])
Saída: [
['X', 's', '-', '#'], ['s', 'X', '-', '#'],
['s', '-', 'X', '#'], ['s', '-', '#', 'X'],
['X', '-', 's', '#'], ['-', 'X', 's', '#'],
['-', 's', 'X', '#'], ['-', 's', '#', 'X'],
['X', '-', '#', 's'], ['-', 'X', '#', 's'],
['-', '#', 'X', 's'], ['-', '#', 's', 'X'],
['X', 's', '#', '-'], ['s', 'X', '#', '-'],
['s', '#', 'X', '-'], ['s', '#', '-', 'X'],
['X', '#', 's', '-'], ['#', 'X', 's', '-'],
['#', 's', 'X', '-'], ['#', 's', '-', 'X'],
['X', '#', '-', 's'], ['#', 'X', '-', 's'],
['#', '-', 'X', 's'], ['#', '-', 's', 'X']
]

- Entrada: (["1", "2", "3"])
Saída: [['1', '2', '3'], ['2', '1', '3'], ['2', '3', '1'], ['1', '3', '2'],
['3', '1', '2'], ['3', '2', '1']]

- Entrada: ([ ])
Saída: [[ ]]

Matemática Classes Estruturas Básicas

21. Sistema de Localização

A estação espacial Olympus, que está nas etapas finais de sua construção, possui um sistema de rastreamento que deve ser capaz de localizar coordenadas próximas a partir da sua própria posição.

Para isso, você foi incumbido da tarefa de construir parte desse sistema.

Você deve construir uma classe capaz de armazenar um ponto de 3 coordenadas numéricas e também detectar o setor em que elas se encontram.

Além disso, também deve ser capaz de calcular a distância entre o ponto e a estação espacial.

Desafio 21: Sistema de Localização

Escreva uma classe que seja capaz de armazenar 3 coordenadas, determinar o setor em que se encontram suas coordenadas e sua distância em relação à estação espacial (coordenada [0, 0, 0]).

Distribuição dos setores:

- Alfa: [positivo, positivo, positivo]
- Beta: [positivo, positivo, negativo]
- Gama: [positivo, negativo, positivo]
- Delta: [positivo, negativo, negativo]
- Épsilon: [negativo, positivo, positivo]
- Zeta: [negativo, positivo, negativo]
- Sigma: [negativo, negativo, positivo]
- Ômega: [negativo, negativo, negativo]

Considere 0 como positivo para garantir que um ponto estará apenas em um único setor.

Testes 21: Sistema de Localização

- Entrada: ([37, 42, 15])
Saída setor: Alfa
Saída distância: 57.948252777780168

- Entrada: ([144, 49, 0])
Saída setor: Alfa
Saída distância; 152.10851389715174

- Entrada: ([-37, 0, 0])
Saída setor: Épsilon
Saída distância: 37

- Entrada: ([-19, -80, -32])
Saída setor: Delta
Saída distância: 88.23264701911646

Strings Manipulação de Tipos

22. Prisão Intergalática

A estação espacial Tartarus é utilizada pela Federação como prisão para criminosos de toda a galáxia e foi construída utilizando a mais avançada tecnologia para evitar qualquer possibilidade de fuga.

Um dos recursos utilizados é o monitoramento constante de cada prisioneiro, oque deve ser feito com cuidado redobrado quando estes são liberados de suas celas.

Sempre que isso ocorre é feito um escaneamento de todos os prisioneiros, que são sempre identificados por números consecutivos e possuem um tempo limite para entrem ou saiam.

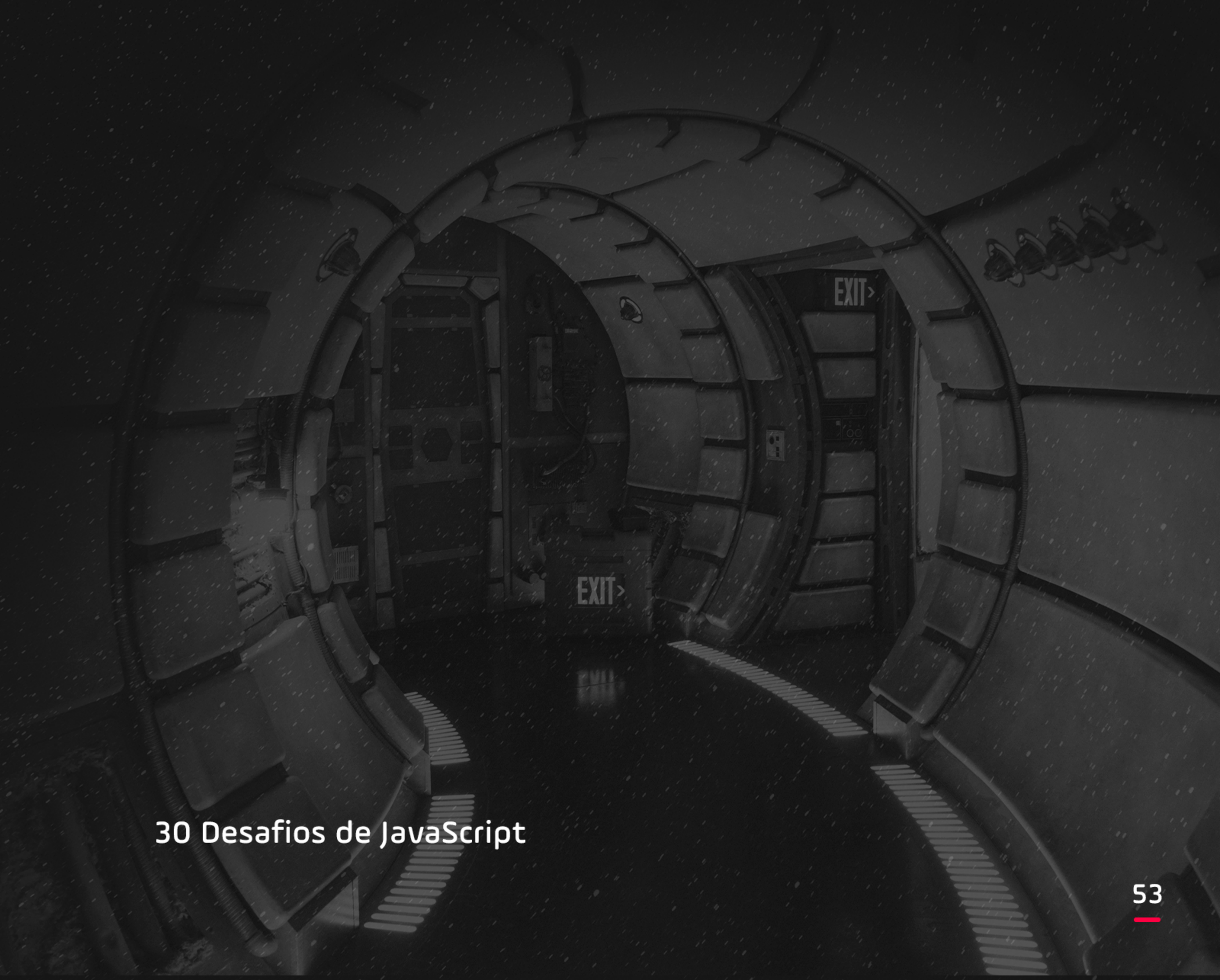
Ao fim desse tempo é gerada uma lista que deve conter os números de todos os prisioneiros, mesmo que em ordem aleatória.

Para garantir que todos entrem e saiam sem que nenhum fique para trás de forma automatizada foi requerido que você construa um código para realizar essa tarefa.

Para esse cenário o último número será um verificador que indicará o número total de prisioneiros, portanto nunca estará faltando.

Desafio 22: Prisão Intergalática

Escreva uma função que receba uma lista embaralhada de números únicos de 1 até n no formato "0001" (string com zeros à esquerda), verifique se há elementos faltando (onde n nunca estará faltando) e, caso hajam, retorne os elementos que faltam.



Testes 22: Prisão Intergalática

- Entrada: (['0020', '0002', '0013', '0004', '0001', '0016', '0015', '0006', '0012', '0007', '0005', '0008', '0011', '0010'])
Saída: ['0003', '0009', '0014', '0017', '0018', '0019']

- Entrada: (['0020', '0009', '0002', '0013', '0004', '0017', '0001', '0003', '0016', '0015', '0019', '0006', '0012', '0007', '0005', '0014', '0008', '0011', '0010', '0018'])
Saída: [ ]

- Entrada: (['0004', '0002', '0005', '0003'])
Saída: ['0001']

- Entrada: ([ ])
Saída: [ ]

PARABÉNS!

Você acaba de concluir
8 desafios Medium e avançar
na sua jornada pela galáxia.

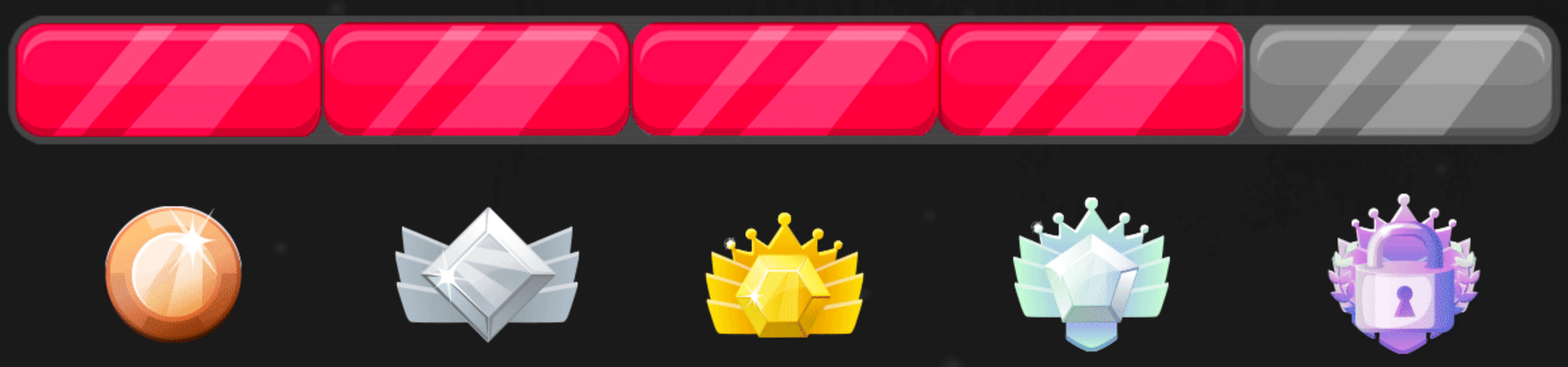
Sua nova atribuição:
Programador(a) JavaScript
das galáxias nível ouro.

onebitcode
FOR AMAZING PROGRAMMERS

Tire um print, publique nos stories do Instagram
e marque @onebitcode

nível

HARD



Strings Arrays

23. Corrida de Pods

Corridas de pods são muito rápidas e perigosas, mas seres de todos os planetas adoram.

Para acompanhar a classificação durante a corrida foi requisitado que você construa um programa que atualize a lista com a posição de todos os corredores a medida que eles vão ultrapassando uns aos outros ou sendo eliminados da corrida.

Desafio 23: Corrida de Pods

Escreva um programa que receba uma lista de classificação de nomes e uma string no formato "Nome +n" (ou -n), onde n é a quantidade de posições para subir ou descer na classificação, e retorne essa mesma lista com a classificação atualizada.

A função também deve ser capaz de receber "Nome ELIMINATE", nesse caso o participante deve ser jogado para o fim da lista e deve ser acrescentado um " ELIMINATED" ao seu nome, indicando que ele foi eliminado.

Os participantes eliminados não podem ter nenhum corredor não eliminado atrás deles na lista.

Assuma que sempre receberá uma entrada válida no formato "Corredor AÇÃO"

Testes 23: Corrida de Pods

- Classificação Inicial: 'Alfa', 'Beta', 'Gama' e 'Delta'

- Entrada: ('Beta +1')
Classificação: 'Beta', 'Alfa', 'Gama' e 'Delta'

- Entrada: ('Gama -1')
Classificação: 'Beta', 'Alfa', 'Delta' e 'Gama'

- Entrada: ('Delta ELIMINATE')
Classificação: 'Beta', 'Alfa', 'Gama' e 'Delta ELIMINATED'

- Entrada: ('Gama +2')
Classificação: 'Gama', 'Beta', 'Alfa' e 'Delta ELIMINATED'

Strings RegEx

24. Verificação de Usuário

Em uma nave de grande porte existem muitos terminais de acesso ao sistema principal, e para se conectar através deles um membro da tripulação deve utilizar seu nome de usuário, senha e verificação biométrica.

O nome de usuário deve ser validado segundo algumas regras antes que possa ser cadastrado.

Sua tarefa atual é construir um mecanismo de validação de nomes de usuário para ser utilizado pelo sistema de cadastro.

Desafio 24: Verificação de Usuário

Escreva uma função que recebe uma string e verifica se ela atende aos seguintes requisitos:

- Deve conter entre 4 e 32 caracteres.
- Deve conter apenas letras (sem acentos), números ou _
- Deve começar com uma letra
- Não pode terminar com _
- Deve conter pelo menos um de cada tipo de caractere (letra, número e underscore)
- Deve ser único

Obs.: Para isso você pode utilizar qualquer meio que achar válido para simular um banco de dados, como um array contendo vários nomes fictícios para comparação, por exemplo.

- Caso atenda, retorne true, caso não atenda retorne false.

Testes 24: Verificação de Usuário

- Usuários Já Registrados: ['erick_14', 'pam_ls2', 'VICTOR_99A']

- Entrada: ('52alfred')
Saída: false

- Entrada: ('erick_14')
Saída: false

- Entrada: ('josh_g15')
Saída: true

- Entrada: ('hugo123_')
Saída: false

- Entrada: ('k_9')
Saída: false

Strings

25. Validações do Compilador

Você e sua equipe estão trabalhando em um compilador que deverá converter o código escrito em uma linguagem moderna para uma outra linguagem mais antiga que é utilizada pelo sistema principal da sua nave.

Uma das tarefas da qual você ficou encarregado foi a de validar o uso de parêntesis, colchetes e chaves no código.

Para que sejam válidos os agrupamentos deve ser abertos e fechados corretamente.

Desafio 25: Validações do Compilador

Escreva uma função que recebe uma string, verifica se ela possui uma distribuição válida de parêntesis, colchetes e chaves, e retorna true ou false baseada nessa verificação.

A função deve ser capaz de funcionar com qualquer string independente do conteúdo que acompanha os parêntesis, colchetes e chaves.

Testes 25: Validações do Compilador

- Entrada: ('((((((([(])))))))')
Saída: false

- Entrada: ('{(){(([]){[]}()()()()[]}(){(())}(())}')
Saída: true

- Entrada: Utilize o código de um desafio que você tenha feito até agora.
Saída: true

- Entrada: Utilize o código de um desafio que você tenha feito até agora com um erro proposital nos parêntesis.
Saída: false

Matemática Recursão

26. Otimização de Cálculos

Para implementar uma determinada operação no computador principal da nave Intrepid está sendo necessário o uso de elementos da sequência de Fibonacci.

Essa sequência é formada por números obtidos através da soma de seus dois anteriores, iniciando-se em 0 e 1.

Você foi designado para criar um procedimento otimizado que seja capaz de encontrar o número de uma dada posição na sequência de Fibonacci.

Desafio 26: Otimização de Cálculos

Escreva uma função que recebe um número e retorna o valor exato da sequência de Fibonacci na posição desse número.

Ela deve ser capaz de funcionar em números grandes como 10000 e retornar números inteiros corretos.

Testes 26: Otimização de Cálculos

- Entrada: (0)
Saída: 0n

- Entrada: (2n)
Saída: 1n

- Entrada: (7)
Saída: 13n

- Entrada: (144n)
Saída: 555565404224292694404015791808n

Recursão Objetos Arrays

27. Encontrando Conexões

Sua equipe recebeu a missão de analisar os dados encontrados em uma base abandonada em um satélite natural do planeta Grenas.

Após fazer o levantamento desses dados eles foram organizados e convertidos para objetos em estrutura de árvore.

Sua tarefa no momento é construir um procedimento que seja capaz de procurar dentro desses objetos por todas as ocorrências de "connection" ou "connections" e encontrar as suas respectivas "_id" e "label".

Desafio 27: Encontrando Conexões

Escreva uma função que receba um objeto e retorne uma lista com os valores de todas as ocorrências da propriedade "_id" e "label" em propriedades "connection" e "connections" onde "connection" é um objeto contendo "_id" e "label" e "connections" é um array de "connection".

Testes 27: Encontrando Conexões

Arquivo data-0.json:

- Entrada: data-0.json
Saída: [
['619459086b56da1078d3cf62', ' pariatur'],
['6194590847e29864f074fd6b', ' minim'],
['6194590836eff3460e579dea', ' ad'],
['6194590809beac7af1db9197', ' velit'],
['61945908daa72c63c52f5917', ' nisi'],
['619459081b9359627ffb174d', ' voluptate'
],
['61945908061bb8557d39c99f', ' aliqua'],
['61945908e0a0b705972e12e6', ' ipsum'],
['6194590890812bf4100f4e9f', ' quis'],
['6194590829921931df2a2d700', ' in'],
['619459088a15378404abf511', ' amet'],
['61945908928aade5d4f39f17', ' laboris'],
['61945908115dc2018ca494a7', ' voluptate'
],
['61945908df0af4bd4a7ab88b', ' amet'],
['6194590898ebfe53111aa2f6', ' eiusmod'],
['6194590860411 3f3e460454d', ' labore'],
['6194590830724 99a9e15c0e5', ' id'],
['619459082b3120ee36002eb1', ' ullamco'
],
['61945908c0026f4a05dfa60c', ' enim'],
['619459082b2187a0757bc3f1', ' consequat'
]
]

- Entrada: data-1.json
Saída: [
['618c602633a65eef8d166a9d', 'incididunt'],
['618c60261ec45f86b0f2a6e1', 'id'],
['618c60267a22bbc164a1ca1f', 'ad'],
['618c60269a345d2c5d093205', 'non'],
['618c6026e00028a0eecb5e7f', 'incididunt'],
['618c6026004b29de6a86c9e3', 'commodo'],
['618c60261f2b40af3f72fc10', 'et'],
['618c6026404cb1f8f8c836c1', 'ea'],
['618c60260ef2ff33ca7c1d57', 'esse'],
['618c6026c0c4bf1f7eceb8e7', 'aliquip'],
['618c60265242a17ce280fd79', 'magna'],
['618c60264aa91981fb641dd6', 'magna'],
['618c6026a5b15da35a1274a0', 'labore'],
['618c60260cf5a8d37599693f', 'et'],
['618c602600906dc4876fb6fc', 'et'],
['618c6026069ca6a43012dcd2', 'magna'],
['618c6026b2cc8c9c0777f86c', 'dolore'],
['618c602638385dd46e7b2956', 'sint'],
['618c602614ac92aaf00bfa3a', 'sit'],
['618c60264898c59023e42240', 'non'],
['618c6026719aefafe40fdc9a', 'exercitation'],
['618c6026bc3b7b80b7bf9931', 'ea'],
['618c60267dbd271dd7e26b8c', 'proident'],
['618c60268a35240a8d75a6c1', 'aliqua'],
['618c60268ff0b056951ec64c', 'qui'],
['618c602606d19ed1320083fe', 'laborum'],

```
[ '618c6026ee0fb0147481dfcd',  'enim' ],
[ '618c602618b35dce16ec05a8',  'nulla' ],
[ '618c6026601a92e297e30a83',  'consectetur' ],
[ '618c6026aa8fb06a41357816',  'exercitation' ],
[ '618c6026a3b67246dc9aa9d5',  'voluptate' ],
[ '618c60265bc857612e17dd2b',  'sint' ],
[ '618c6026635822092780f8f4',  'duis' ],
[ '618c6026cfff4c7e1cfa65ae',  'sunt' ],
[ '618c6026206b642b514488de',  'ut' ],
[ '618c6026552023c63bed8bde',  'aute' ],
[ '618c60263a6d2833ff4b864c', 'mollit' ],
[ '618c60263144b79ff8078216', 'exercitation' ],
[ '618c60264368c05e182edd60', 'elit' ],
[ '618c6026839e89a98572d747', 'sunt' ],
[ '618c60264a5f8510f5cdd3c3', 'anim' ],
[ '618c6026972cf7622a8c106e', 'eu' ],
[ '618c6026f11cd25fe384265c', 'ea' ],
[ '618c60263e2b23d7dd3eac0e', 'ut' ],
[ '618c60265228eea004c0bfe5', 'non' ],
[ '618c6026070ddf3e45991960', 'nostrud' ],
[ '618c6026f57d56275b589150', 'ipsum' ],
[ '618c60269b45de4e20a3404b', 'dolor' ],
[ '618c6026526cba6891ed5fb9', 'magna' ],
[ '618c60262e9c33e82c1d770c', 'labore' ]
]
```

Classes Objetos Arrays Matemática

28. Hora de Jogar

Há exatamente 100 anos, quando a exploração de novos planetas ainda era algo reservado apenas para as ficções-científicas, um RPG de mesa muito popular te permitia viver aventuras em outros planetas como um explorador do espaço.

Agora, para comemorar essa data especial uma equipe está criando uma versão digital deste mesmo jogo e você foi designado para cooperar com ela no desenvolvimento.

Sua tarefa no momento é criar, de acordo com uma lista de requisitos, uma classe que representa um explorador e que será incluída no jogo final.

É obrigatório o uso de classes.

Desafio 28: Hora de Jogar

Escreva uma classe que calcula e mantém informações sobre exploradores, como seus nível e habilidades. Ela precisará obedecer aos seguintes requisitos:

Sobre o nível:

- Um explorador começa no nível 1 e pode evoluir até o nível 99.

- O explorador não deve subir de nível após o nível 99, mas pode acumular pontos de experiência.

- Para subir de nível o explorador deve acumular 100 pontos + 10 pontos * o seu nível atual para cada nível (ex.: Nv. 1: 110 pts, Nv.2: 120 pts, Nv. 3: 130 pts, etc)

- Para aumentar o seu nível um explorador deve ganhar pontos de experiência através da ação de explorar.

- Os pontos de experiência devem se manter acumulados mesmo após subir de nível.

Sobre o ranqueamento:

Um explorador deve ser ranqueado entre 6 ranques diferentes: "Novato", "Explorador", "Veterano", "Elite", "Mestre", "Lenda" (ou qualquer nomenclatura que preferir)

O ranqueamento deve obedecer à seguinte ordem:

- 1-9: Novato
- 10-29: Explorador
- 30-49: Veterano
- 50-79: Elite
- 80-98: Mestre
- 99: Lenda

Sobre a ação de explorar:

- A ação de explorar precisa de um planeta a ser explorado.

- O planeta a ser explorado precisa ter um id, um nome, um nível de hostilidade (entre pacífico, neutro e hostil) e um tipo de terreno (ex.: desértico, florestal, montanhoso, subaquático, etc).

- Um explorador morto não pode explorar.

- Deve ser possível saber todos os planetas que um explorador já explorou com sucesso.

Sobre o resultado:

A ação de explorar pode ser bem sucedida ou falhar. Para determinar o resultado vai ser preciso simular um "rolar de dados", sorteando dois números aleatórios entre 1 e 6, com resultados entre 2 e 12.

Resultados entre 5 e 12 garantem sucesso em planetas pacíficos.

Resultados entre 7 e 12 garantem sucesso em planetas neutros.

Resultados entre 9 e 12 garantem sucesso em planetas hostis.

Resultado 2 (dois números 1) em planetas hostis o explorador morre (mas não deve ser excluído).

Após 3 resultados 12 (dois números 6) em planetas de um mesmo terreno o explorador se torna um especialista naquele terreno e recebe um bônus de +1 no resultado dos dados.

Esse bônus impede que ele morra em caso de falha crítica e aumenta as chances de sucesso.

Só é possível acumular esse bônus uma vez.

Sobre os pontos de experiência

Em caso de sucesso na ação de explorar o Explorer deve receber pontos de experiência de acordo com o seguinte:

- 15 pts - Planeta pacífico
- 25 pts - Planeta neutro
- 50 pts - Planeta hostil

Em caso de falha na ação de explorar o Explorer deve receber pontos de experiência de acordo com o seguinte:

- 0 pts - Planeta pacífico
- 0 pts - Planeta neutro
- 10 pts - Planeta hostil

Testes 28: Hora de Jogar

Assuma que temos um objeto explorador está no nível 9 e possui 1340 pontos de experiência.

Assuma também que ele já obteve 2 acertos críticos (6+6) explorando terrenos do tipo 'forest'.

Seu ranque deve ser "Novato"

Assuma que ele explorou um novo planeta { id: 1, name: 'Planeta 1', hostility: 'neutral', terrain: 'forest' }

Ao obter sucesso nessa exploração ele deve:

- Ganhar 25 pontos de exp.
- Avançar para o nível 10.
- Avance para o ranque Explorador.
- Se torne especialista em terrenos 'forest' e ganhe o bônus de +1
- Ter o planeta de 'Planeta 1' na sua lista de planetas conhecidos.

Assuma agora que ele explorou o planeta { id: 2, name: 'Planeta 2', hostility: 'hostile', terrain: 'desert' } e tirou dois 1 no resultado

Ao obter uma falha crítica nessa exploração ele deve:

- Morrer, se tornando incapaz de explorar novamente.

- O objeto desse explorador ainda deve conter as suas informações.

PARABÉNS!

Você acaba de concluir
6 desafios Hard e avançar
na sua jornada pela galáxia.

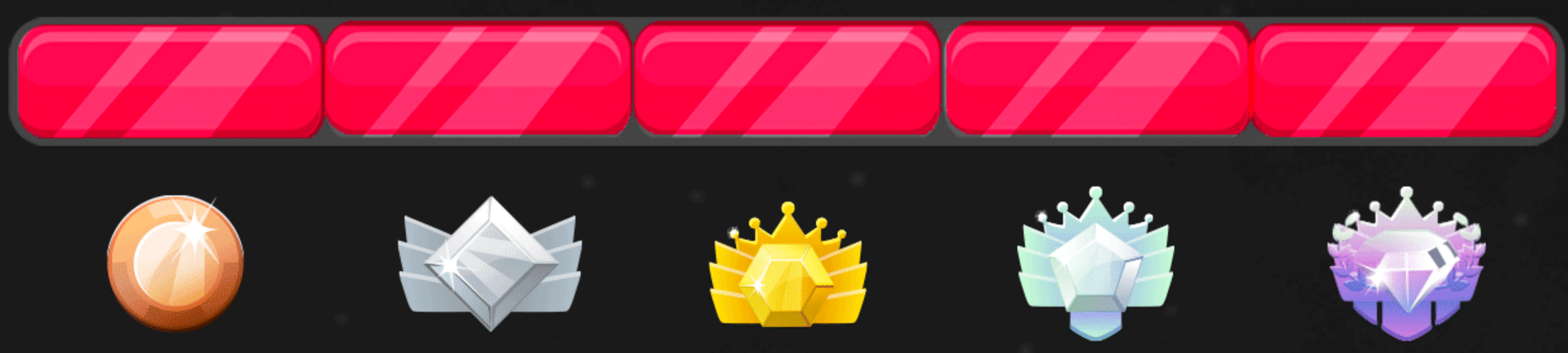
Sua nova atribuição:
Programador(a) JavaScript
das galáxias nível platina.

onebitcode
FOR AMAZING PROGRAMMERS

Tire um print, publique nos **stories do Instagram**
e marque @onebitcode

nível

VERY HARD



Arrays Classes Estruturas de Dados

29. Labirintos Subterrâneos

Os túneis subterrâneos utilizados pela raça nativa do planeta Krawl funcionam como labirintos de diversos andares para impedir que forasteiros encontrem sua cidade escondida.

Você e sua equipe possuem o mapeamento dos túneis, mas agora precisam de uma forma de saber se é possível chegar de um ponto a outro sem ficarem presos pelas paredes dos túneis.

Além disso, os túneis não são seguros, sendo crucial avançar rapidamente, logo vocês também precisam descobrir qual é o caminho mais rápido para se chegar da entrada até a saída de cada andar.

Para isso você precisa desenvolver um programa que leia o mapa do túnel, representado por um array bidimensional, e verifique qual é o caminho mais rápido entre a entrada e a saída.

Desafio 29: Labirintos Subterrâneos

Escreva um programa capaz de receber uma matriz bidimensional composta de caminhos livres, bloqueados, a entrada e a saída do labirinto, e verifique se é possível chegar do início até o fim apenas utilizando movimentos horizontais e verticais e também qual é o caminho mais curto para isso.

A representação dos valores será feita com espaços (" "), tralhas ("#"), um começo ("S") e um fim ("E"), mas fique a vontade para nomeá-las como preferir.

Testes 29: Labirintos Subterrâneos

- Entrada: [
  ['#', ' ', ' ', ' ', ' ', '#', ' '],
  [' ', 'S', ' ', ' ', '#', ' ', ' '],
  [' ', ' ', '#', ' ', ' ', '#', ' '],
  [' ', '#', ' ', ' ', ' ', '#', 'E'],
  [' ', ' ', ' ', ' ', ' ', ' ', ' '],
  ['#', '#', ' ', ' ', '#', ' ', ' '],
  [' ', ' ', ' ', ' ', ' ', ' ', ' '],
]
Saída: ['(1, 1)', '(1, 2)', '(1, 3)', '(2, 3)', '(3, 3)', '(4, 3)', '(4, 4)', '(4, 5)', '(4, 6)', '(3, 6)']

- Entrada: [
  ['E', ' ', ' ', ' '],
  ['#', ' ', '#', ' '],
  [' ', ' ', ' ', ' '],
  [' ', '#', '#', ' '],
  [' ', ' ', '#', ' '],
  ['#', ' ', '#', ' '],
  [' ', ' ', '#', ' '],
  [' ', ' ', ' ', 'S'],
]
Saída: ['(7, 3)', '(6, 3)', '(5, 3)', '(4, 3)', '(3, 3)', '(2, 3)', '(1, 3)', '(0, 3)', '(0, 2)', '(0, 1)', '(0, 0)']

- Entrada: [
  [' ', ' ', ' ', ' ', ' '],
  [' ', ' ', 'S', ' ', ' '],
  [' ', '#', '#', ' ', ' '],
  [' ', '#', 'E', '#', ' '],
  [' ', '#', ' ', '#', ' '],
  [' ', ' ', ' ', '#', ' '],
  [' ', ' ', ' ', ' ', ' '],
]
Saída: ['(1, 2)', '(1, 1)', '(1, 0)', '(2, 0)', '(3, 0)', '(4, 0)', '(5, 0)', '(5, 1)', '(5, 2)', '(4, 2)', '(3, 2)']

- Entrada: [
  [' ', '#', '#', 'S', '#', '#', '#'],
  ['#', '#', ' ', ' ', ' ', ' ', '#'],
  ['#', ' ', ' ', '#', '#', ' ', '#'],
  ['#', ' ', ' ', ' ', ' ', ' ', '#'],
  ['#', ' ', '#', ' ', '#', ' ', '#'],
  ['#', ' ', ' ', ' ', '#', ' ', '#'],
  ['#', '#', ' ', '#', ' ', ' ', '#'],
  ['#', '#', ' ', ' ', ' ', '#', '#'],
  ['#', ' ', ' ', ' ', '#', ' ', '#'],
  ['#', ' ', '#', ' ', ' ', '#', '#'],
  ['#', ' ', ' ', '#', '#', '#', '#'],
  ['#', ' ', ' ', ' ', ' ', ' ', '#'],
  ['#', '#', ' ', ' ', '#', ' ', '#'],
  ['#', '#', '#', '#', '#', 'E', '#'],
]
Saída: ['(0, 3)', '(1, 3)', '(1, 2)', '(2, 2)', '(2, 1)', '(3, 1)', '(4, 1)', '(5, 1)', '(5, 2)', '(6, 2)', '(7, 2)', '(8, 2)', '(8, 1)', '(9, 1)', '(10, 1)', '(11, 1)', '(11, 2)', '(11, 3)', '(11, 4)', '(11, 5)', '(12, 5)', '(13, 5)']

Manipulação de Arquivos

Matemática Arrays Strings Classes Promises

30. Planilha Portátil

Um colega da equipe de tecnologia ficou encarregado de construir um pequeno dispositivo portátil capaz de trabalhar com planilhas e cálculos.

Ele é habilidoso com microcontroladores, porém pediu a sua ajuda para desenvolver o programa que irá lidar com as planilhas.

Ele precisa que você crie uma pequena aplicação capaz de ler e escreverem uma planilha, tenha algumas funções matemáticas básicas e consiga abrir e salvar arquivos de planilhas.

Desafio 30: Planilha Portátil

Escreva uma aplicação que simula o funcionamento de uma planilha.

Ela precisa armazenar um array bidimensional de células e possuir no mínimo quatro funções principais: leitura de célula, escrita de célula, salvar em arquivo e abrir de um arquivo.

A leitura de uma célula deve receber o seu nome, que é uma string como "A2", "C4" ou "J1" onde a letra é a coluna e o número é a linha, e retornar o valor da célula.

A escrita de uma célula pode receber o seu nome e um valor, e deverá guardar esse valor na célula.

As fórmulas devem ser strings no formato FORMULA(células), onde as células podem ser selecionadas individualmente ou em grupo através dos caracteres ; e : respectivamente.

Por exemplo, A2;A5 seria A2 E A5, já A2:A5 seria A2 ATÉ A5.

Para ler e salvar em arquivos é possível utilizar o módulo "fs" incluído por padrão no Node.js.

Ao ler um arquivo o programa deve carregar a planilha para uso como no momento em que ela foi salva, mas a forma como ela será armazenada em arquivo é livre.

A escrita de uma célula também pode receber o seu nome e uma fórmula.

Nesse caso, deverá executar a fórmula e guardar o resultado na célula.

Devem estar disponíveis pelo menos as seguintes fórmulas (use os nomes que achar melhor):

SUM: soma os valores das células passadas.
Pode receber qualquer quantidade de células.

SUB: subtrai os valores das células passadas.
Pode receber qualquer quantidade de células, mas deve subtrair da primeira todos os outros valores.

MUL: multiplica os valores das células passadas.
Pode receber qualquer quantidade de células.

DIV: divide os valores das células passadas.
Pode receber qualquer quantidade de células, mas deve dividir o valor da primeira pelas seguintes.

MIN: retorna o menor valor dentre as células passadas.
Pode receber qualquer quantidade de células.

MAX: retorna o maior valor dentre as células passadas.

AVG: retorna a média aritmética dos valores das células passadas. Pode receber qualquer quantidade de células.
Pode receber qualquer quantidade de células.

Testes 30: Planilha Portátil

Testes (em sequência na mesma planilha):

- Escrever o valor 4 na célula 'A1'
  Escrever o valor 4 na célula 'A2'
  Escrever o valor 0 na célula 'B1'
  Escrever o valor 3 na célula 'B2'
  Escrever o valor 0 na célula 'C1'
  Escrever o valor 3 na célula 'C2'
  Escrever a fórmula 'SUM(A1:C2)' na célula 'A3'
  A célula 'A3' deve conter o valor 14

- Escrever o valor 8 na célula 'A1'
  A célula 'A3' deve conter o valor 18

- Escrever o valor 'AVG(A1;A2;A3)' na célula 'A4'
  A célula 'A4' deve conter o valor 10

- Escrever o valor 9 na célula 'B1'
  Escrever o valor 1 na célula 'B3'
  Escrever a fórmula 'SUB(B1:B3)' na célula 'B4'
  A célula 'A3' deve conter o valor 27
  A célula 'A4' deve conter o valor 13
  A célula 'B4' deve conter o valor 5

- Escrever o valor 'MIN(A1:A5)' na célula 'A6'
  Escrever o valor 'MAX(B1:B5)' na célula 'B6'
  A célula 'A6 deve conter o valor 4
  A célula 'B6' deve conter o valor 9

- Escrever o valor 18 na célula 'C1'

Escrever a fórmula 'DIV(C1;C2)' na célula 'C3'

A célula 'A3' deve conter o valor 45

A célula 'A4' deve conter o valor 19

A célula 'C3' deve conter o valor 6

- Escrever o valor 4 na célula 'D1'

Escrever o valor 8 na célula 'D2'

Escrever a fórmula 'MUL(D1;D2)' na célula 'D3'

Escrever a fórmula 'SUM(D1:D3)' na célula 'D4'

Escrever a fórmula 'SUM(A1:D4)' na célula 'D5'

A célula 'D3' deve conter o valor 32

A célula 'D4' deve conter o valor 44

A célula 'D5' deve conter o valor 209

- Salvar o arquivo com um nome qualquer

Abrir o arquivo em uma planilha diferente (os dois existirão simultaneamente)

Escrever o valor 'Teste' na célula 'G1' da segunda planilha

Escrever o valor 3 na célula 'G2' da segunda planilha

Escrever o valor 4 na célula 'G3' da segunda planilha

Escrever a fórmula 'SUM(G2;G3)' na célula 'G4' da segunda planilha

A segunda planilha (lida do arquivo) deve conter todos os valores da primeira

A célula 'G1' deve conter o valor 'Teste' apenas na segunda planilha

A célula 'G2' deve conter o valor 3 apenas na segunda planilha

A célula 'G3' deve conter o valor 4 apenas na segunda planilha

A célula 'G4' deve conter o valor 7 apenas na segunda planilha

PARABÉNS!

Você acaba de concluir
2 desafios Very Hard e avançar
na sua jornada pela galáxia.

Sua nova atribuição:
Programador(a) JavaScript
das galáxias nível diamante.

onebitcode
FOR AMAZING PROGRAMMERS

Tire um print, publique nos stories do Instagram
e marque @onebitcode

onebitcode{}
FOR AMAZING PROGRAMMERS

JORNADA COMPLETA

Você completou
30 desafios de JavaScript
do nível Very Easy ao Very Hard

Agora está pronto(a) para desenvolver projetos F#D@S com JavaScript!

Tire um print, publique nos **stories do Instagram**
e marque @onebitcode