

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' ],
[ '619459082921931df2a2d700', ' in' ],
[ '619459088a15378404abf511', ' amet' ],
[ '61945908928aade5d4f39f17', ' laboris' ],
[ '61945908115dc2018ca494a7', ' voluptate' ],
[ '61945908df0af4bd4a7ab88b', ' amet' ],
[ '6194590898ebfe53111aa2f6', ' eiusmod' ],
[ '61945908604113f3e460454d', ' labore' ],
[ '619459083072499a9e15c0e5', ' 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' ],
[ '618c6026cff4c7e1cfa65ae', '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**