Atividade 2

DESCRIÇÃO

Cada vez mais os processos seletivos estão utilizando resolução de problemas para analisar o nível de compreensão dos conceitos de programação do candidato. Para simular esse processo será exigido a entrega de no **mínimo quatro** dos algoritmos descritos a seguir.

QUESTÕES

QUESTÃO 1 - Crie um algoritmo que receba um texto e retorne a soma do código de cada caractere do texto. Então se o texto possuir apenas um caractere será retornado o seu código, ou seja, em `a` retorna-se `97`. Já quando o texto possui mais de um caractere será somado o código cada caracter, por exemplo, no texto `ifpb` os códigos dos caracteres `i, f, p, b` são `105, 102, 112, 98`, cuja a soma equivale à `417`.

Para analisar mais exemplos veja a *Tabela 1.1*.

Tabela 1.1:

Entrada	Saída
`a`	`97`
`b`	`98`
`A`	`65`
`ifpb`	`417`

`lorem`	`543`
`lorem ipsum`	`1133`

QUESTÃO 2 - Desenvolva uma ferramenta para contabilizar a quantidade de Pokemons um jogador falta adquirir em sua Pokedex. Então, recebe a lista de Pokemons e retorne dos 151 disponíveis quantos faltam ser capturados (Fonte: Urionlinejudge - Pomekon Collection).

Por exemplo, se um jogador capturou os Pokemons `Zubat, Pikachu, Pikachu`, então, ao retirar a duplicidades percebe-se que faltam capturar `149` Pokemons, que seria `total - capturado` ou nesse caso `151 - 2`.

Para analisar mais exemplos veja a *Tabela 2.1*.

Tabela 2.1:

Entrada	Saída
`Zubat`	`150`
`Zubat, Pikachu, Pikachu`	`149`
`Zubat, Zubat, Zubat, Zubat, Zubat, Zubat, Zubat, Zubat`	`150`
`Zubat, Charmander, Caterpie, Pidgeot`	`147`
`Charmander, Caterpie, Pidgeot, Rattata, Zubat, Zubat, Zubat`	`146`

QUESTÃO 3 - O Sr. Severino resolveu organizar sua biblioteca, então ele pensou em criar um programa que registrasse e classificasse seus livros pelo seu código (Fonte: Urionlinejudge - The Library of Mr. Severino).

Então considerando que fosse coletado os códigos `3000, 2000, 1000` a ordem da classificação seguirá a ordem numerica por meio do resultado `1000, 2000, 3000`.

Para analisar mais exemplos veja a *Tabela 3.1*.

Tabela 3.1:

Entrada	Saída
`3000, 2000, 1000`	`1000, 2000, 3000`
`1233, 0015, 0100`	`0015, 0100, 1233`
`0752, 1110, 0001, 6322, 8000, 6321, 0000`	`0000, 0001, 0752, 1110, 6321, 6322, 8000`

QUESTÃO 4 - Na casa de Sam existe uma árvore de maçã e de laranja. Para contabilizar quantas frutas estavam caindo em sua casa ele criou um programa que identifica a exata posição em que as frutas caiam.

Para auxiliar na contagem ele criou uma escala que determina a posição da árvore de maçã (`a`), da casa que compreende o intervalo entre os pontos `s` e `t`, e por fim a posição da árvore de laranja (`b`) (Fonte: Hackerrank - Apple and Orange).

Vale frisar que os sensores das frutas são posicionados nas árvores o que implica que as coordenadas possuíram o seu referencial, ou seja, ao receber as coordenadas das maçãs pelas posições `-2, 2, 1` significa que a primeira caiu à esquerda da árvore enquanto que as duas seguintes ficaram à direita.

Entretanto para determinar se as frutas caíram na casa, no intervalo de `s` até `t`, é preciso referenciar as posições das frutas pelas coordenadas de `a` e `b`.

Então considerando inicialmente os valores de `a`, `s`, `t` e `b` como sendo `5`, `10`, `0` e `15`, já as posições maçãs como sendo `0, 3, 6`, e das laranjas `17, 20`:

O resultado é que as maçãs baseadas na referência da posição `0` serão convertidos em `0+0, 0+3, 0+6`, já a posição da laranja `15` converterá as posições como sendo `15+17, 15+15`.

Quanto comparamos as posições das maçãs e laranjas, respectivamente `0, 3, 6` e `32, 30`, com o intervalo da casa entre `5` e `10` percebe-se que `1` maçã e `0` laranjas caíram na casa:

Para auxiliar na calibragem do programa veja mais exemplos na *Tabela 4.1* considerando as entradas `a`, `s`, `t`, `b`, `coordenada das maçãs` e `coordenada das laranjas`. O resultado final deve contabilizar quantas maçãs e laranjas caíram na casa.

Tabela 4.1:

Entrada	Saída
`5`, `10`, `0`, `15`, `0, 3, 6`, `17, 20`	`1, 0`
`7`, `11`, `5`, `15`, `-2, 2, 1`, `5, -6`	`1, 1`
`8`, `9`, `-1`, `15`, `-2, 2, 10`, `20, 21`	`1, 0`

QUESTÃO 5 - Considere que dois gatos e um rato estão posicionados em cima de um muro, dado as posições dos dois gatos e o rato calcule qual gato pegará o rato primeiro considerando que ambos se movimentam a mesma velocidade constante. Entretanto, se os dois gatos chegaram no mesmo instante no rato, ele irá conseguir escapar pois os gatos começaram a brigar (Fonte: Hackerrank - Cats and a Mouse).

Para auxiliar o cálculo as coordenadas `A`, `B`, `C` serão relacionados respectivamente para o `Gato A`, o `Gato B`, o rato, ou seja, se recebermos as coordenadas `1, 2, 3` vejam que as distâncias do `Gato A` e `Gabo B` para o rato serão `2` e `1`, o que significa que a saída será `Cat B` pois a distância dele é menor:

Entretanto, nas coordenadas `1, 3, 2` os gatos possuem uma distância de `1`, o que significa que eles vão se estranhar o rato saíra ileso, resultando na saída `Mouse C`:

Para analisar mais exemplos veja a *Tabela 5.1*, contudo veja que as possibilidades de saída são `Cat A`, `Cat B` e `Mouse C`.

Tabela 5.1:

Entrada	Saída
`1, 2, 3`	`Cat B`
`1, 3, 2`	`Mouse C`
`1, 4, 2`	`Cat A`

QUESTÃO 6 - Você está no comando do bolo para o aniversário da sua sobrinha e decidiu que o bolo terá uma vela para cada ano de sua idade total. Quando ela soprar as velas, ela só poderá soprar as mais altas. Sua tarefa é descobrir quantas velas ela poderá soprar (Fonte: Hackerrank - Birthday Cake Candles).

Considerando que sua sobrinha vai completar 4 anos com as velas de tamanho `1, 1, 1, 3`, veja que a mais alta é a vela de altura `3` o que significa que do total de velas ela só poderá apagar `1` vela. Já se a altura das velas fossem `1, 3, 1, 3` como a vela mais alta ainda continua sendo `3` será possível apagar `2` velas.

Para analisar mais exemplos veja a *Tabela 6.1*.

Tabela 6.1:

Entrada	Saída
`1, 1, 1, 3`	`1`

`1, 3, 1, 3`	`2`
`1, 3, 3, 3`	`3`
`3, 2, 1, 3`	`2`
`18, 90, 90, 13, 90, 75, 90, 8, 90, 43`	`5`

QUESTÃO 7 - Gary está precisando obter um controle preciso sobre a ocorrência de vales sobre um relevo. Para auxiliar essa detecção foi criado um sistema que conseguia detectar as subidas (uphill - U) e descidas (downhill - D) no caminhada de um percurso (Fonte: Hackerrank - Counting Valleys).

Então, se o sensor detectar esse movimento `DU` significar que houve uma descida seguida de uma subida resultando em `1` vale:

Já no movimento `DUDU` foi detectado `2` vales:

Contudo, um vale só pode ser contabilizado quando a descida passa do nível zero e em seguida volta ao nível zero, por exemplo no percurso `UDDDUDUU` existe apenas `1` vale:

Para analisar mais exemplos veja a *Tabela 7.1*.

Tabela 7.1:

Entrada	Saída
`DU`	`1`
`DUDU`	`2`
`UUUDU`	`0`

`UDDDUDUU`	`1`
`DDUUDDUDUUUD`	`2`

QUESTÃO 8 - Crie um programa para detectar se um texto possui caracteres suficientes para montar a palavra `hackerrank`na ordem original dos caracteres (Fonte: Hackerrank - HackerRank in a String).

Por exemplo, para o texto `hereiamstackerrank` é possível verificar que é existem caracteres suficientes para montar a palavra deseja conforme as letras em destaque `HereiAmstaCKERRANK`. Nessa situação o programa deve retorna `YES`, caso contrario o retorno será `NO`.

Para analisar mais exemplos veja a *Tabela 8.1*.

Tabela 8.1:

Entrada	Saída
`hereiamstackerrank`	`YES`
`hackerworld`	`NO`
`hhaacckkekraraannk`	`YES`
`rhbaasdndfsdskgbfefdbrsdfhuyatrjtcrtyytktjjt`	`NO`

QUESTÃO 9 - Crie um programa que compare duas cadeias de três números, `A` e `B`, par a par para determinar quantos são os valores de `A` maiores que `B` e vice-versa (Fonte: Hackerrank - Compare the Triplets).

Por exemplo, seja `A` e `B` carregados com os valores `1, 1, 1` e `0, 0, 0` significa que o resultado será que `1 > 0`, `1 > 0` e `1 > 0`, ou seja, em `A` os `3` valores são maiores que `B`, e `0` elementos de `B` são maiores que `A`.

Para analisar mais exemplos veja a *Tabela 9.1*.

Tabela 9.1:

Entrada	Saída
`1, 1, 1`, `0, 0, 0`	`3, 0`
`0, 0, 0`, `1, 1, 1`	`0, 3`
`17, 28, 30`, `99, 16, 8`	`2, 1`
`5, 6, 7`, `3, 6, 10`	`1, 1`

QUESTÃO 10 - Tente determinar se um texto é divertido ou não. Para determinar essa característica é preciso obter o código do texto na ordem direta e inversa (Fonte: Hackerrank - Funny String).

Por exemplo, no texto `abc` o primeiro passo do processo é gerar o seu inverso, que no caso seria `cba`, o próximo passo será gerar o código de cada caractere. No exemplo `abc` e `cba` seriam `97, 98, 99` e `99, 98, 97`.

Entretanto, para determinar se o texto é divertido ou não é necessário gerar a diferença absoluta de cada caractere dois a dois, no exemplo na direta e indireta seria respectivamente `97-98, 98-99` e `99-98, 98-97`. O resultado de ambos será `1, 1`, o que retornará o valor `Funny`.

No texto `abd`, os códigos seriam `97, 98, 100` e `100, 98, 99`, resultando nas diferenças de `1, 2` e `2, 1`, ou seja, como as diferenças não são iguais o retorno será `No Funny`.

Para analisar mais exemplos veja a *Tabela 10.1*.

Tabela 10.1:

Entrada	Saída
`abc`	`Funny`

`abd`	`Not Funny`
`acxz`	`Funny`
`bcxz`	`Not Funny`