Módulo Recursividade

Curso Estruturas de Dados e Algoritmos Expert

Prof. Dr. Nelio Alves
https://devsuperior.com.br

Lista de exercícios

Soluções:
https://github.com/devsuperior/curso-eda/tree/main/recursividade

Problema "soma-naturais"
Faça uma função que, dado um número natural N, retorne a soma dos números de 0 até N.

Exemplo 1:

Entrada	**Saída**
`0`	`0`

Exemplo 2:

Entrada	**Saída**
`2`	`3`

Exemplo 3:

Entrada	**Saída**
`4`	`10`

```
Assinaturas:

Javascript:
function sumNaturals(n)

Java:
public static int sumNaturals(int n)

C#:
public static int SumNaturals(int n)

Python:
def sum_naturals(n)
```

Problema "fatorial"
O fatorial de um número natural N é a multiplicação de 1 até N, exceto para o valor 0 (zero), cujo fatorial por definição é 1. Faça uma função para retornar o fatorial de um dado número.

Exemplo 1:

Entrada	**Saída**
`0`	`1`

Exemplo 2:

Entrada	**Saída**
`3`	`6`

Exemplo 3:

Entrada	**Saída**
`4`	`24`

```
Assinaturas:

Javascript:
function factorial(n)

Java:
public static int factorial(int n)

C#:
public static int factorial(int n)

Python:
def factorial(n)
```

Problema "fibonacci"
A sequência de Fibonacci começa com 0, 1, e depois cada número é a soma de seus dois antecessores: 0 1 1 2 3 5 8 13...
Faça uma função para retornar o valor de uma dada posição da sequência de Fibonacci.

Exemplo 1:

Entrada	**Saída**
`0`	`0`

Exemplo 2:

Entrada	**Saída**
`0`	`1`

Exemplo 3:

Entrada	**Saída**
`6`	`8`

```
Assinaturas:

Javascript:
function fib(n)

Java:
public static int fib(int n)

C#:
public static int fib(int n)

Python:
def fib(n)
```

Problema "reverse"

Faça uma função que receba uma lista (de qualquer tipo) e retorne a lista reversa.

Exemplo 1:

Entrada	**Saída**
`[]`	`[]`

Exemplo 2:

Entrada	**Saída**
`["azul"]`	`["azul"]`

Exemplo 3:

Entrada	**Saída**
`["azul", "verde", "preto", "rosa"]`	`["rosa", "preto, "verde", "azul"]`

Assinaturas:

Javascript:
```
function reverse(list)
```

Java:
```
public static <T> List<T> reverse(List<T> list)
```

C#:
```
public static List<T> reverse<T>(List<T> list)
```

Python:
```
def reverse(lst)
```

Problema "potencia"
Escreva uma função recursiva que calcule a potência de um número dado uma base A e um expoente B, ambos inteiros. Lembre-se que qualquer número elevado ao expoente 0 é igual a 1.

Exemplo 1:

Entrada	**Saída**
{ "a": 5, "b": 0 }	1

Exemplo 2:

Entrada	**Saída**
{ "a": 3, "b": 4 }	81

Assinaturas:

Javascript:
```
function power(a, b)
```

Java:
```
public static int power(int a, int b)
```

C#:
```
public static int power(int a, int b)
```

```
Python:
def power(a, b)
```

Problema "contagem-regressiva"
Implemente uma função recursiva que exiba uma contagem regressiva de um número natural qualquer até zero. A função não deve retornar nada, ou seja, apenas imprima os valores no console.

Exemplo:

Entrada	**Saída (impressão no console)**
5	5 4 3 2 1 0

```
Assinaturas:

Javascript:
function countdown(n)

Java:
public static void countdown(int n)

C#:
public static void Countdown(int n)

Python:
def countdown(n)
```

Problema "soma-lista"
Crie uma função recursiva que retorne a soma de todos os elementos em uma lista de números. Se a lista for vazia, a função deve retornar o valor 0 (zero).

Exemplo 1:

Entrada	**Saída**
[]	0

Exemplo 2:

Entrada	**Saída**
`[4, 5, 3]`	`12`

Assinaturas:

Javascript:
```
function sumList(list)
```

Java:
```
public static double sumList(List<Double> list)
```

C#:
```
public static double SumList(List<double> list)
```

Python:
```
def sum_list(lst)
```

Problema "menor-elemento"
Faça uma função recursiva para retornar o menor elemento de uma lista de números. Supor que a lista tenha pelo menos um elemento.

Exemplo:

Entrada	**Saída**
`[10, 15, 20, 8, 30, 17]`	`8`

Assinaturas:

Javascript:
```
function minor(list)
```

Java:
```
public static double minor(List<Double> list)
```

C#:
```
public static double Minor(List<double> list)
```

Python:
```
def minor(lst)
```

Problema "mesclar-listas"
Faça uma função recursiva que mescle os elementos de duas listas A e B, retornando a lista resultante. As listas A e B podem ser de qualquer tipo, mas as listas A e B devem ser do mesmo tipo. As listas não precisam ser do mesmo tamanho.

Exemplo 1:

Entrada	**Saída**

{
  "a": [10, 20, 30],
  "b": [5, 8, 7]
}
	[10, 5, 20, 8, 30, 7]

Exemplo 2:

Entrada	**Saída**

{
  "a": ["ana", "maria"],
  "b": ["joao", "bob", "alex", "leo"]
}
	["ana", "joao", "maria", "bob", "alex", "leo"]

Assinaturas:

Javascript:
```
function mergeLists(a, b)
```

Java:
```
public static <T> List<T> mergeLists(List<T> a, List<T> b)
```

C#:
```
public static List<T> MergeLists<T>(List<T> a, List<T> b)
```

Python:
```
def merge_lists(a, b)
```

Problema "checar-ordenação"
Faça uma função recursiva para verificar se uma dada lista de números está ordenada.

Exemplo 1:

Entrada	**Saída**
[]	true

Exemplo 2:

Entrada	**Saída**
`[15, 20, 22, 31, 40]`	`true`

Exemplo 3:

Entrada	**Saída**
`[15, 20, 22, 21, 40]`	`false`

Assinaturas:

Javascript:
```
function isSorted(list)
```

Java:
```
public static boolean isSorted(List<Double> list)
```

C#:
```
public static bool IsSorted(List<double> list)
```

Python:
```
def is_sorted(lst)
```

Problema "conta-caracteres"

Crie uma função recursiva que conte quantas vezes um caractere específico aparece em uma string. O resultado não deve diferenciar maiúsculas de minúsculas. Para evitar consumo excessivo de memória, sua implementação não deve instanciar substrings da string original durante o processo.

Dica 1: utilize uma função recursiva auxiliar com um parâmetro adicional correspondente à posição que estiver sendo percorrida na string.

Dica 2: sua função auxiliar pode ainda ser com recursividade de cauda.

Exemplo:

Entrada	**Saída**

`{`
`  "ch": 'b',`
`  "text": "Batata para o bebê"`
`}`
	`3`

Assinaturas:

Javascript:
```
function charCount(ch, text)
```

```
Java:
public static int charCount(char ch, String text)

C#:
public static int charCount(char ch, string text)

Python:
def char_count(ch, text)
```

Problema "palindromo"

Um texto é palíndromo se seu inverso é igual ao texto original. Crie uma função para determinar se uma data string é um palíndromo. Utilize uma função auxiliar com recursividade de cauda com parâmetros adicionais para evitar a necessidade de se instanciar substrings repetidamente nas chamadas recursivas.

Exemplo 1:

Entrada	**Saída**
""	true

Exemplo 2:

Entrada	**Saída**
"aba"	true

Exemplo 3:

Entrada	**Saída**
"abccba"	true

Exemplo 4:

Entrada	**Saída**
"abcfba"	false

```
Assinaturas:

Javascript:
function isPalindrome(text)

Java:
public static boolean isPalindrome(String text)
```

C#:
```
public static bool IsPalindrome(string text)
```

Python:
```
def is_palindrome(text)
```