Lista de Exercícios 02
Professor: Mauricio de Souza

1. **Multiplicação Escalar em Lista**: Crie uma função multiplica_escalar(lista, escalar) que recebe uma lista de números e um escalar, e multiplica cada elemento da lista pelo escalar, retornando a nova lista.
2. **Produto de uma Lista**: Escreva uma função produto_lista(lista) que calcula o produto (multiplicação) de todos os elementos de uma lista.
3. **Conta Consoantes**: Implemente uma função conta_consoantes(texto) que conta o número de consoantes em uma string.
4. **Concatenação de Listas**: Crie uma função concatena_listas(lista1, lista2) que recebe duas listas e retorna uma nova lista com os elementos de ambas as listas concatenados.
5. **Soma dos N primeiros Naturais**: Escreva uma função soma_naturais(n) que retorna a soma dos N primeiros números naturais.
6. **Diferença de Conjuntos**: Implemente uma função diferenca_conjuntos(lista1, lista2) que retorna os elementos presentes em lista1 que não estão em lista2.
7. **Ordenação com Selection Sort**: Crie uma função selection_sort(lista) que ordena uma lista usando o algoritmo de Selection Sort.
8. **Ordenação com Insertion Sort**: Implemente uma função insertion_sort(lista) que ordena uma lista usando o algoritmo de Insertion Sort.
9. **Ordenação com Quick Sort**: Escreva uma função quick_sort(lista) que ordena uma lista usando o algoritmo de Quick Sort (versão não recursiva).
10. **Diferença entre Máximo e Mínimo**: Crie uma função diferenca_max_min(lista) que retorna a diferença entre o maior e o menor elemento de uma lista.
11. **Produto Escalar entre Vetores**: Implemente uma função produto_escalar(vetor1, vetor2) que retorna o produto escalar entre dois vetores (listas de números) de mesmo tamanho.
12. **Listas Pares e Ímpares**: Escreva uma função separa_pares_impares(lista) que separa uma lista de números em duas listas: uma com os pares e outra com os ímpares.
13. **Diferença Absoluta entre Elementos de Lista**: Crie uma função diferença_elementos_lista(lista) que retorna uma lista com a diferença absoluta entre cada par consecutivo de elementos da lista original.
14. **Desvio Padrão de Lista**: Implemente uma função desvio_padrao(lista) que calcula o desvio padrão dos elementos em uma lista de números.
15. **Multiplicação de Matrizes 2x2**: Crie uma função multiplica_matrizes(matriz1, matriz2) que recebe duas matrizes 2x2 e retorna o resultado da multiplicação entre elas.
16. **Busca Binária**: Implemente uma função busca_binaria(lista, elemento) que realiza uma busca binária em uma lista ordenada e retorna o índice do elemento, ou -1 se não for encontrado.

17. **Intercala Listas Ordenadas**: Crie uma função intercala_listas_ordenadas(lista1, lista2) que recebe duas listas ordenadas e retorna uma nova lista intercalada e ordenada.
18. **Média Ponderada**: Escreva uma função media_ponderada(lista_valores, lista_pesos) que calcula a média ponderada de uma lista de valores com uma lista de pesos correspondente.
19. **Diagonais de uma Matriz**: Implemente uma função diagonais_matriz(matriz) que recebe uma matriz quadrada e retorna uma lista contendo os elementos das duas diagonais.
20. **Conta Números em Intervalo**: Crie uma função conta_intervalo(lista, inicio, fim) que conta quantos números em uma lista estão dentro de um intervalo [inicio, fim].
21. **Converte Celsius para Fahrenheit**: Escreva uma função celsius_para_fahrenheit(celsius) que converte uma temperatura em Celsius para Fahrenheit.
22. **Converte Fahrenheit para Celsius**: Crie uma função fahrenheit_para_celsius(fahrenheit) que converte uma temperatura em Fahrenheit para Celsius.
23. **Multiplicação de Valores Pares**: Implemente uma função multiplica_pares(lista) que multiplica todos os números pares em uma lista.
24. **Números Primos até N**: Escreva uma função numeros_primos(n) que retorna uma lista de todos os números primos até um número n dado.
25. **Concatenação de Strings**: Crie uma função concatena_strings(lista_strings) que recebe uma lista de strings e retorna uma única string com todas as strings concatenadas.
26. **Ordenação com Merge Sort**: Implemente uma função merge_sort(lista) que ordena uma lista usando o algoritmo de Merge Sort.
27. **Verifica Ordem Crescente**: Crie uma função esta_ordenada(lista) que verifica se uma lista está ordenada em ordem crescente.
28. **Remove Múltiplos de um Número**: Escreva uma função remove_multiplos(lista, n) que remove todos os múltiplos de n de uma lista.
29. **Intervalo entre Elementos de uma Lista**: Implemente uma função intervalo_entre_elementos(lista) que calcula o intervalo (diferença) entre o maior e o menor valor de uma lista.
30. **Histograma de Ocorrências em Lista**: Crie uma função histograma(lista) que recebe uma lista e retorna um dicionário com o número de ocorrências de cada elemento da lista.

May the force be whit you