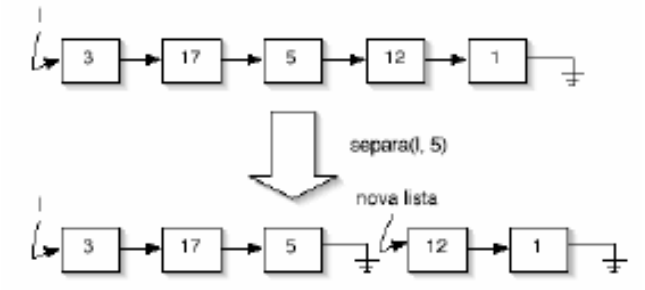
# Estrutura de Dados I – Lista de Exercícios

Sugestão: Fazer exercícios sobre listas encadeadas do site HackerRank
https://www.hackerrank.com/domains/data-structures

Considere uma SLL, DLL, CSLL e DSLL e escreva um algoritmo para executar cada uma das seguintes operações:

1. Inserir um novo dado na lista:
    a. Na primeira posição
    b. Na última posição da lista.
    c. Na posição k-ésima da lista se esta posição existir
    d. Após um nó identificado por uma chave key
    e. Antes de um nó identificado por uma chave key
    f. Insere um nó identificado por uma chave key em um lista ordenada.
2. Remover um nó da lista
    a. Da primeira posição
    b. Da última posição da lista.
    c. Da k-ésima posição da lista se esta posição existir
    d. Nó identificado por uma chave key
    e. Após um nó identificado por uma chave key
    f. Antes de um nó identificado por uma chave key
3. Concatenar duas listas.
4. Liberar todos os nós numa lista.
5. Inverter uma lista de modo que o último elemento se torne o primeiro, e assim por diante.
6. Combinar duas listas ordenadas numa única lista ordenada.
7. Formar uma lista contendo a união dos elementos de duas listas.
8. Formar uma lista contendo a intersecção dos elementos de duas listas.
9. Retornar a soma dos dados numa lista.
10. Retornar o número de elementos numa lista.
11. Criar uma segunda cópia de uma lista.
12. Verificar se duas listas ligadas dadas são iguais.
13. Receber como parâmetro uma lista e um valor inteiro n e divida a lista em duas, de tal forma que a segunda lista comece no primeiro nó logo após a primeira ocorrência de n na lista original. A figura a seguir ilustra essa separação. A função deve retornar um ponteiro para a segunda sub-divisão da lista original, enquanto l deve continuar apontando para o primeiro elemento da primeira subdivisão da lista.



14. Receber uma lista l e um número inteiro n, e remove da lista os n primeiros nós. Caso n seja maior que o número de nós da lista remove todos os nós e deixa a lista vazia.
15. Detectar um ciclo em uma lista.
16. Remover nós com dados duplicados de uma determinada lista vinculada.
17. Remover nós consecutivos com dados duplicados de uma determinada lista vinculada.
18. Retornar o k-ésimo nó contado a partir do último elemento da lista.
19. Verificar se a lista é um palíndromo.
20. Receber uma lista e dividir seus nós para criar duas listas menores. As listas menores devem ser criadas a partir de elementos alternados na lista original. Portanto, se a lista original for {a, b, a, b, a}, então uma sub-lista deverá ser {a, a, a} e a outra deverá ser {b, b}.
    a. Fazer alocando novos nós
    b. Fazer sem alocar novos nós
21. Encontrar o elemento do meio de uma lista
22. Inserir um dado no meio de uma lista
23. Calcular o comprimento de uma lista L1:
24. Receber duas listas (L1 e L2) e retornar um valor lógico: Verdadeiro, se L1 = L2 e Falso, se L1 <> L2 (mesmos dados).
25. Receber duas listas (L1 e L2) e retornar uma cópia L2 da lista L1;
26. Receber duas listas (L1 e L2) e retornar uma lista L igual à diferença L1-L2;
27. Receber duas listas (L1 e L2) e retornar um valor inteiro igual ao número de valores comuns às duas listas L1 e L2;
28. Receber uma lista L e eliminar os nós de ordem par (segundo, quarto, sexto, etc.)
29. Receber uma lista L e eliminar os nós de ordem impar (primeiro, terceiro, quinti, etc.)
30. Receber uma lista L e reorganizar seus nós em duas listas: <primeiro, terceiro, quinto, ...> e <segundo≥, quarto, sexto ≥, ...>. Não aloque novos nós.
31. Receber uma lista L e dois valores de chave (key1 e key2) e trocar a posição dos nós que contém key1 e key2 entre sí . Trocar a posição dos nós.
32. Receber duas listas (L1 e L2) e criar uma terceira lista L3 que represente:
    a. L3 é a união de l1 e l2
    b. L3 é a interseção de l1 e l2
33. Receber duas listas (L1 e L2) e Retorna Verdadeiro se L1 esta contido em L2 e falso caso contrário.