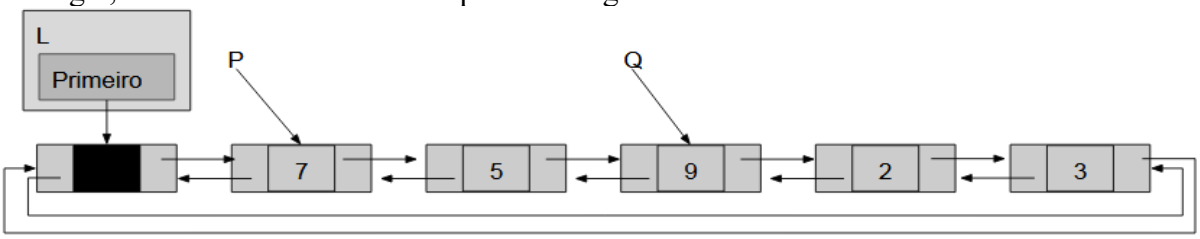
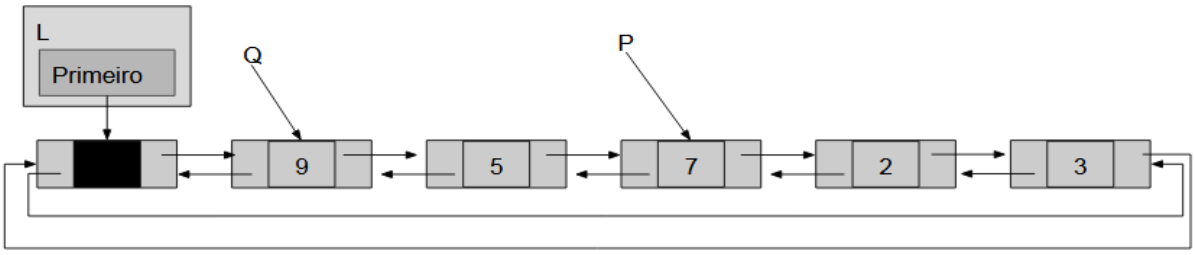
Lista de Exercícios

1. Faça um operação que verifique se uma lista L está com suas chaves em ordem crescente, sendo L:
   a) Uma lista ligada sem sentinela e não circular.
   b) Duplamente ligada circular com sentinela.
2. Faça uma operação que receba uma lista duplamente ligada circular com sentinela e retorne o seu tamanho.
   Exemplo:
   L=[1,6, 5, 1, 7, 0, -3, 22]
   Tamanho(L) = 8
3. Faça uma operação que receba uma lista duplamente ligada circular com sentina e retorne a maior chave que existe na lista.
4. Faça uma operação que receba uma lista duplamente ligada circular com sentinela e retorne a menor chave que existe na lista.
5. Faça uma operação que receba uma lista duplamente ligada circular com sentinela e retorne a soma das chaves de todos elementos da lista.
6. Implemente uma operação trocar, que dois ponteiros para duas células das lista e os troque de lugar, considerando uma lista duplamente ligada circular com sentinela.



trocar(L, P, Q)



Obs: Existem alguns casos distintos:
Quando P é sucessor imediato de Q;
Quando Q é sucessor imediato de P;
Quando P está “antes” de Q; e
Quando Q está “antes” de P.

7. Utilizando as operações já implementadas, faça um subprograma que calcule a média das chaves de todos elementos da lista.
8. Faça um subprograma que receba uma lista, L1, e retorne uma cópia desta lista, usando apenas as operações definidas para lista.
9. Faça um subprograma que receba uma lista L1 e retorne uma cópia desta lista, porém ordenada por chave.
10. Faça um subprograma que receba duas listas, L1 e L2, e retorne uma terceira lista, L3 que é a concatenação entre L1 e L2.
    Exemplo:
    L1 = [1, 5, 2, 6, 9] e L2 = [1, 7, 2, 3]
    L3 = [1, 5, 2, 6, 9, 1, 7, 2, 3]

11. Faça um subprograma, chamado merge, que receba duas listas ordenadas e retorne uma terceira lista também ordenada.
    Exemplo:
        L1=[1, 3, 7, 9, 12] e L2 = [2, 5, 10, 15, 21, 33]
        merge(L1,L2) (ou merge(L3,L1,L2)) = [1, 2, 3, 5, 7, 9, 10, 12, 15, 21, 33]
12. Faça um subprograma que receba duas listas, L1 e L2, e retorne uma lista L3 contendo:
    a) A união entre L1 e L2.
    b) A intersecção entre L1 e L2.
    Obs: Use apenas operações sobre listas para criar L3.
13. Escreva um subprograma que filtre uma lista, ou seja, dado uma lista e uma chave ch, todos elementos com aquela chave devem ser removidos da lista.
    Exemplo:
        L = [1, 5, 6, 4 ,2, 1, 4 ,3]
        filtra(L, 4) = [1, 5, 6, 2, 1, 3]
14. Escreva um subprograma que receba uma lista L e remova todos elementos repetidos de L.
    Exemplo:
    L = [1, 5, 6, 4 ,2, 1, 4 ,3]
    exclui_repetidos(L) = [1, 5, 6, 4, 2, 3]
15. Faça um subprograma que receba uma lista L e um valor inteiro N. Seu subprograma deverá separar a L em 2 listas (L1, e L2), no ponto N. Se N for maior ou igual ao tamanho de L, L1 devera conter os mesmos elementos de L e L2 deverá ser uma lista vazia.
    Exemplo1:
        L=[1, 2, 6, 7, 3, 8, 15, 22]
        Separa(L,L1, L2,3)
        L=[1, 2, 6, 7, 3, 8, 15, 22]
        L1 = [1, 2, 6]
        L2 = [7, 3, 8, 15, 22]
    Exemplo 2:
        L=[1, 2, 6, 7, 3, 8, 15, 22]
        Separa(L,L1, L2,8)
        L=[1, 2, 6, 7, 3, 8, 15, 22]
        L1 = [1, 2, 6, 7, 3, 8, 15, 22]
        L2 = [ ] // lista vazia
16. Dado o agregado heterogêneo aluno, implemente uma aplicação para manipular duas turmas (listas de alunos) com esse tipo de dado com as seguintes operações:

```
typedef struct aluno{
        int ra;
        char nome[100];
}aluno;
```

    a) Inserir aluno na turma;
        - Deve-se solicitar qual turma o aluno deverá ser inserido (0 ou 1);
        - Se um ra de aluno existir na turma 1, não poderá ser permitido inserir um aluno na turma 2 com nome diferente e mesmo ra;
    b) Remover aluno da turma;
    c) Listar todos alunos de uma turma
    d) Verificar alunos matriculados em ambas as turmas;
        - Deverão ser mostrados somente os alunos que estão tanto na turma 1 quanto na turma 2;