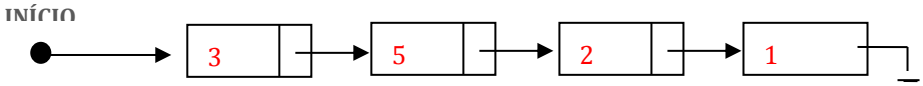
EXERCÍCIOS COM LISTAS ENCADEADAS

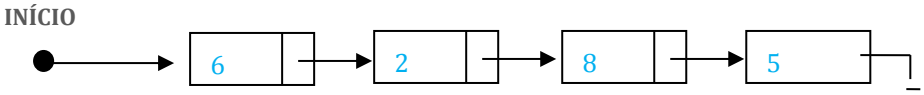
1. Dadas duas listas genéricas, escreva uma função que produza uma nova lista composta pela **união** das listas (eliminando as repetições), conforme exemplo ilustrado a seguir (o qual usa números inteiros para facilitar o entendimento do mesmo):

struct DLista* uniao (struct DLista* lista1, struct DLista* lista2, funcaoComparacao fc);

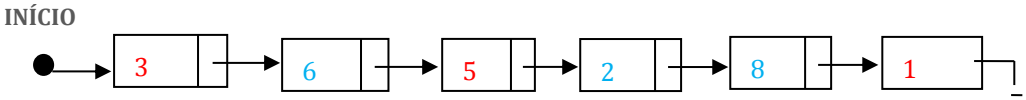
lista 1



lista 2



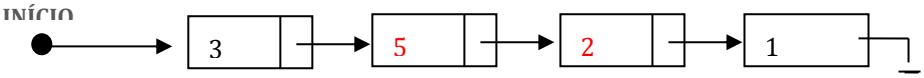
Nova Lista



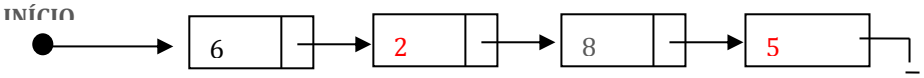
2. Dadas duas listas genéricas, escreva uma função que produza uma nova lista somente com as informações que aparecem em ambas as listas (interseção), conforme exemplo ilustrado a seguir (também baseado em números inteiros para facilitar o entendimento):

struct DLista* intersecao (struct DLista* lista1, struct DLista* lista2, funcaoComparacao fc);

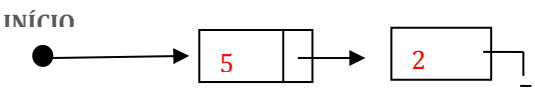
lista 1



lista 2



Nova Lista



3. Escreva uma função para determinar se duas listas são **disjuntas**, ou seja, se as listas não possuem nenhuma informação igual.

int disjunta (struct DLista* lista1, struct DLista* lista2, funcaoComparacao fc);

4. Escreva uma função para determinar se uma lista está **contida** em outra lista.

int contida (struct DLista * lista1, struct DLista * lista2, funcaoComparacao fc);

5. Escreva uma função para ordenar uma lista encadeada em ordem crescente (o algoritmo de ordenação usado não é relevante para a solução deste exercício).

void ordenar (struct DLista * lista, funcaoComparacao fc);

6. Escreva uma função **RECURSIVA** para imprimir as informações contidas em uma lista encadeada.

void imprimir (struct DLista * lista, funcaoImpressao fi);

7. Escreva uma função para incluir uma informação em uma lista encadeada em uma posição específica.

struct DLista * incluir (struct DLista * lista, void * info, int pos);

8. Escreva duas versões da função para inverter uma lista encadeada, isto é, o primeiro é trocado com o último, o segundo é trocado com o penúltimo e assim por diante.

   Nesta versão, a inversão acontece na lista original.

void inverteLista (struct DLista * lista);

   Nesta outra versão, a lista original não é alterada, isto é, invertida. O resultado da inversão gerará uma nova lista encadeada.

struct DLista *inverteLista (struct DLista * lista);

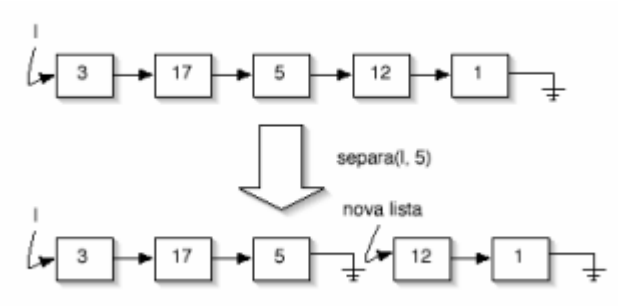
9. Implemente uma função que receba como parâmetro um vetor e o tranforma em uma lista encadeada, conforme o protótipo a seguir:

struct DLista * constroiLista (void * vet[ ], int tam);

10. Implemente também a operação inversa:

void* constroiVetor (struct DLista * lista);

11. Considerando uma lista genérica, implemente uma função que receba como parâmetro uma lista encadeada e uma informação qualquer (void * info) e divida a lista em duas, de tal forma que a segunda lista comece no primeiro nó logo após a primeira ocorrência de "**info**" na lista original.
    A figura a seguir ilustra como deveria ser feita esta separação (considerando como exemplo uma lista de números inteiros).



    Essa função deve obedecer ao seguinte protótipo:

    struct DLista * divideLista (struct DLista * lista, void * info, funcaoComparacao fc);

    A função deve retornar um descritor para a segunda lista (a segunda parte da lista resultante da divisão da lista original), enquanto o descritor passado como parâmetro deve continuar apontando para o primeiro elemento (da primeira parte resultante da divisão) da lista original.

12. Implemente um programa em C que gerencie um evento esportivo de corrida de rua. As informações relevantes para cada competidor são: *o número de inscrição*, *apelido*, *idade* e o *tempo de prova*.
    O programa deve ter funcionalidades para gerenciar a lista de competidores (inclusão e exclusão de competidores), alteração dos dados dos competidores (especificamente o tempo de prova) e o ranking da corrida (por categoria, geral e acima de 60 anos).

    O objetivo deste exercício é apenas usar a biblioteca de listas encadeadas. Caso seja necessário fazer eventuais ajustes na biblioteca, os mesmos podem ser feitos.