Listas ligadas simples

A. Considere as seguintes definições dos tipos de dados **INFO**, **Nodo** e **PNodo** (apontamentos das aulas teóricas):

```
typedef struct {
   int  NFatura;        // chave com valores entre 1000 e 9000
   int  NIF;            // valores entre 100000 e 200000
   int  data[3];        // dia-mês-ano com o ano com valores entre 2010 e 2022
   float  Pagamento;    // valores entre 1.0 e 1000.0 (euros)
} INFO;

struct Nodo {
   INFO  Elemento;
   struct Nodo *Prox;
};
typedef struct Nodo *PNodo;
```

Cada elemento (registo) do tipo **INFO** corresponde a uma compra efetuada num estabelecimento comercial. Assim, o campo **NFatura** deve ser único (chave).

Copiar as seguintes bibliotecas:

Aleatorio (**.h** e **.c**) ---> que contém operações para gerar números aleatoriamente

OperacoesPrimariasA (**.h** e **.c**) ---> que contém definida a estrutura **INFO**

ListasLigadasSimples (**.h** e **.c**) ---> que contém definidas a estrutura **Nodo** e o tipo **PNodo**

que se encontram disponíveis na página web da disciplina (**Folhas práticas ---> Bibliotecas e Exercícios das folhas práticas ---> Listas ligadas simples**).

Elaborar um programa em C que utilize as operações contidas nas bibliotecas referidas e resolva as questões colocadas a seguir, acrescentado-as uma a uma ao programa. Usar o programa **ListasExA.c**, que se encontra na página web da disciplina (ver em *bibliotecas ...*).

1. Implementar uma função que crie uma lista **L** com **N** elementos do tipo **INFO** gerados aleatoriamente. Para tal, usar o gerador de números inteiros e reais (biblioteca **Aleatorio**).
   Usando a função anterior, construa uma lista com **N** elementos (**0 ≤ N ≤ 15**) do tipo **INFO**.
2. Mostrar a lista criada antes, em que os elementos são apresentados
   a) do início para o fim,
   b) do fim para o ínicio.
3. Implementar uma função iterativa para determinar a soma dos valores pagos (campo **Pagamento**) de todos os elementos de uma lista **L** cujo campo **NIF** é igual a dado valor **Num**.
4. Implementar uma função recursiva para determinar a quantidade de elementos duma lista **L** com valores do campo **Pagamento** maiores ou iguais a um dado valor **Pag**.

5. Implementar uma função (iterativa ou recursiva), que receba uma lista **L** com elementos do tipo **INFO** e determine (e devolva) o **maior** valor no campo *Pagamento* dos elementos da lista **L**.

6. Implementar uma função (iterativa ou recursiva) que determine (e devolva) o menor elemento duma lista **L** com elementos do tipo **INFO**, com valor no campo **Pagamento** maior ou igual a um dado valor **K**.

7. Implementar uma função que, dados uma lista **L** com elementos do tipo **INFO** e um inteiro **Num**, remova o elemento da lista **L** com valor no campo **NFatura** igual a **Num**.

8. Implementar uma função que remova todos os elementos de uma lista **L** com elementos do tipo **INFO**, cujos elementos sejam **menores** do que o **maior** valor do campo **Pagamento** da lista **L** (usar a função implementada num exercício anterior).

9. Implementar uma função que, dado uma lista **L** com elementos do tipo **INFO** e um inteiro **Num**, remova todos os elementos da lista **L** com valor no campo **NIF** igual a **Num**.

10. Implementar uma função que, dado uma lista **L** com elementos do tipo **INFO** e um número inteiro positivo **N**, remova da lista **L** os seus **N** primeiros elementos e devolva a lista resultante. Caso **N** seja maior do que o comprimento da lista, todos os seus elementos devem ser removidos e o resultado deve ser uma lista vazia.

11. Implementar uma função que receba como parâmetros uma lista **L** com elementos do tipo **INFO** e um número inteiro positivo **Num**, e divida a lista em duas, de tal forma que a segunda lista comece com o elemento que se segue ao primeiro elemento com valor no campo **NFatura** igual a **Num**.

12. Implementar um função que como parâmetro uma lista **L** com elementos do tipo **INFO**, e ordene-a por ordem crescente do campo **NIF**.

13. Implementar uma função que, dado uma lista **L** com elementos do tipo **INFO** ordenados crescentemente pelo campo **NIF** e um valor inteiro **Num**, remova todos os elementos da lista **L** com valores no campo **NIF** iguais a **Num**, caso existam.

14. Implementar uma função que, dado uma lista **L** com elementos do tipo **INFO** ordenados crescentemente pelo campo **NIF** e um inteiro **X**, remova todos os elementos da lista **L** com valores no campo **NIF** superiores ou iguais a **X**, caso existam.

15. Implementar uma função para determinar o número de elementos duma lista **L** com elementos do tipo **INFO**, cujos valores do campo **Pagamento** são maiores que **K1** e menores que **K2**.

16. Implementar uma função para alterar a lista **L** com elementos do tipo **INFO**, tal que aos valores do campo **Pagamento** dos elementos da primeira metade da lista **L** são acrescentados o valor **K1** e aos restantes são acrescentados o valor **K2**.

17. Implementar uma função para alterar uma lista **L** com elementos do tipo **INFO**, tal que ao valor do campo **Pagamento** dos primeiros e últimos **K** elementos da lista **L** são acrescentados o valor **Y1** e aos restantes são acrescentados o valor **Y2**.

18. Construa uma função para determinar o maior elemento duma lista **L** com elementos do tipo **INFO***,* com maior valor no campo **Pagamento**, mas menor do que o valor **Y**.

19. Implementar uma função que dada uma lista **L** com elementos do tipo **INFO**, determine o elemento da lista com o segunfo maior valor do campo **Pagamento**, percorrendo a lista apenas uma vez.

20. Implementar uma função que dada uma lista **L** com elementos do tipo **INFO**, remova o primeiro elemento e o último elemento da lista **L**.

21. Implementar uma função que dada uma lista **L** com elementos do tipo **INFO**, remova todos os elementos de **L** com exceção do primeiro elemento e do último elemento.

22. Implementar uma função que dada uma lista **L** com elementos do tipo **INFO**, passe o último elemento para a segunda posição da lista **L**.

23. Implementar uma função que destrua uma dada lista **L** com elementos do tipo **INFO** (liberte todos os nodos da lista **L**, que fica vazia).

24. Implementar uma função que dada uma lista **L** com elementos do tipo **INFO**, inverta a lista **L** de modo que o último elemento se torne o primeiro, e assim sucessivamente.

25. Implementar uma função que dada uma lista **L** com elementos do tipo **INFO** e um número inteiro **N**, elimine o **N-ésimo** elemento da lista **L**.

26. Implementar uma função que dadas duas listas, **L1** e **L2**, com elementos do tipo **INFO** ordenadas pelo campo **NFatura**, junte as duas listas numa única lista ordenada.

B. Considere a seguinte definição do tipo de dados **INFO**:

```
typedef struct{
   int  NCC;            // chave e com valores entre 100000 e 900000
   int  dataNasc[3];    // dia-mês-ano com ano entre 1920 e 2022
   float  altura;       // em metros
   int  genero;         // 0 = Feminino e 1 = Masculino
}INFO;
```

Elaborar um programa que utilize as operações que se encontram nas bibliotecas criadas antes e resolva as questões colocadas a seguir, uma a uma.

1. Implementar uma função que receba como parâmetro um inteiro **N** e crie uma lista **L** com N elementos do tipo **INFO** gerados aleatoriamente; para tal, use o gerador de números inteiros que se encontram na biblioteca "Aleatorio.h" (ver página web da disciplina).

2. Mostrar a lista criada antes, em que os elementos são apresentados
   a) do início para o fim, e
   b) do fim para o ínicio.

3. Implementar uma função para determinar a quantidade de elementos de uma lista **L**, cujo valor do campo **altura** é maior ou igual a um dado valor **A** ($A \in [1.0, 2.0]$).

4. Implementar uma função que, dados uma lista **L** com elementos do tipo **INFO** e um inteiro A (A em {1920, …, 2022}), determine e devolva a quantidade de elementos nascidos no ano **A**.

5. Implementar uma função que, dados uma lista **L** com elementos do tipo **INFO** e um número inteiro positivo **N** (N em {1, …, 20}), remova da lista **L** os **N** primeiros elementos de L e devolva a lista resultante. Caso **N** seja maior do que o comprimento da lista, todos os seus elementos devem ser removidos e o resultado deve ser uma lista vazia.

6. Implementar uma função que dado uma lista **L** com elementos do tipo **INFO,** construa 2 listas do mesmo tipo (INFO), uma com todos os elementos do género Masculino (campo *genero* = 0) e a outra com todos os elementos do género Feminino (campo *genero* = 1).

C. Outras problemas

1. Implementar uma função para verificar se um número inteiro positivo **N** é capicua (número que ao ser lido da esquerda para a direita ou vice-versa é o mesmo; ex: 12321), usando uma lista ligada.

2. Implementar uma função que dadas duas lista ligadas com valores binários (0/1), **L1** e **L2**, e um valor inteiro **k** (k > 0), construa duas listas ligadas **L3** e **L4**, em que **L3** é formado pelos primeiros **k** valores de **L1** e os últimos de **L2**, e **L4** é formado pelos primeiros **k** valores de **L2** e os últimos de **L1** (ex: **L1** = [1 1 0 1 0 1], **L2** = [0 1 0 1 1 0] e **k = 2** => **L3** = [1 1 0 1 1 0] e **L4** = [0 1 0 1 0 1]).

3. Seja um polinómio de grau n:

$$P(x) = a_n x^n + a_{n-1} x^{n-1} + ... + a_1 x^1 + a_0 x^0$$

onde $a_n \neq 0$, exceto possivelmente no caso n = 0.

a) Construa uma função para determinar o valor de P(x).

b) Construa uma função para efetuar a soma de dois polinómios: P(x) + P(y).

c) Construa uma função para efetuar o produto de dois polinómios: P(x) x P(y).

Sugestão: Criar uma lista ligada com três campos, um para o coeficiente (a_n), outro para o expoente (n) e o outro para a ligação ao próximo elemento.