Exercícios sobre LISTASLINEARES:

Para a implementação dos exercícios propostos, utilize uma das bibliotecas **lista_cf.h ou listaligada.h** (veja no site da disciplina). No início do programa acrescente um include com o caminho onde encontrar este arquivo. Essa biblioteca, foi construída especialmente para os testes dos exercícios, para que o aluno se acostume com a ideia de abstração.

Para os exercícios , NÃO UTILIZE RECURSOS DE C++, os únicos aceitos são:

- comandos: cin e cout;
- passagem por referência;

Use a definição e as operações como definidas anteriormente:
tipo: Lista
operações: init(L)
inserir(L,v,i),
v = eliminar(L,i),
consultar(L,i),
a = vazia(L),
c = compr(L),
alterar(L,v,i).

Ordem:
include <....>

typedef int elemento_lista;
#***include*** “lista_cf.h” ou “listaligada.h”

1. Construiu uma função que faz a leitura de vários números inteiros e os armazena numa **lista linear**.
2. Construiu uma função que faz a imprime os elementos de uma **lista linear** de inteiros.
3. Construir uma função que recebe uma **lista linear** L, com números inteiros, e verifica se os elementos da lista estão em ordem crescente. O resultado deverá retornar através de um return. Na main, imprimir mensagem.
4. Dadas duas **listas linear** L1 e L2, contendo elementos quaisquer. Supor que a primeira lista comporta ambas. Construir uma função que acrescenta a lista L2 no final da L1.
5. Dadas duas **listas linear** L1 e L2, ambas contendo números inteiros ordenados. Construir uma função que cria uma terceira **lista linear** L3 contendo os elementos da interseção de L1 com L2.
6. Dada uma **lista linear** L, contendo números inteiros quaisquer. Construir uma função que elimina dessa lista o maior elemento.
7. Dadas duas **listas linear** L1 e L2, contendo números inteiros ordenados. Construir uma função que cria uma terceira **lista linear** L3 contendo os elementos de L1 que não pertencem à interseção de L1 com L2.

8. Construir uma função que recebe como parâmetro uma **lista linear** contendo números inteiros (entre1 e 10), não ordenados, e que há várias repetições de todos eles. Construir uma segunda lista para conter: na 1ª posição a quantidade de repetições do número 1, na 2ª as repetições do número 2, e assim por diante, até a 10ª posição.
9. Construir uma função que recebe como parâmetro uma **lista linear** contendo números inteiros quaisquer e que altera cada elemento multiplicando-o por 2.
10. Construir uma função que cria uma **lista linear** com a seguinte regra: o 1º elemento deverá ser igual a 1, o 2º é igual ao primeiro*2, o 3º igual ao segundo *2, e assim por diante. Parar quando o cálculo do elemento for maior do que 1000 e neste caso não inserir o resultado maior do que 1000.
11. Construir uma função que recebe como parâmetro uma **lista linear** contendo números inteiros ordenados e verifica se há algum número repetido na lista ou não. Caso exista alguma repetição retornar 1 senão retornar 0.
12. Dada uma **lista linear** contendo números inteiros quaisquer. Construir uma função que reorganiza a lista colocando os números pares no início da lista e os ímpares no final. O algoritmo deve ser feito apenas com a lista linear dada.
13. Construir uma função que cria uma **lista linear** para conter em cada posição um termo da seguinte soma:

$$\frac{1}{1} + \frac{3}{2} + \frac{5}{3} + \frac{7}{4} + \ldots + \frac{99}{50}$$

14. Construir uma função que cria uma **lista linear** com 50 elementos, onde cada elemento da lista é um termo da seguinte série:

$$\frac{1!}{1} - \frac{2!}{3} + \frac{3!}{5} - \frac{4!}{7} + \frac{5!}{9} - \frac{6!}{11} + \frac{7!}{13} - \frac{8!}{15} \ldots\ldots$$

Isto é, o termo: $\frac{1!}{1}$ é o primeiro elemento da lista; $\frac{-2!}{3}$ é o segundo, e assim por diante.

15. Dada uma **lista linear** contendo números inteiros **desordenados.** Construir uma função que verifica se há algum número repetido na lista ou não. Caso exista alguma repetição retornar 1 senão retornar 0. Não usar estruturas auxiliares para a solução, tais como, vetor ou outra lista linear.
16. Dada uma **lista linear** contendo números inteiros quaisquer. Construir uma função que verifica qual o número que se repete mais vezes nessa lista. Retornar a porcentagem de repetições desse número em relação à lista toda. Não há restrição quanto ao uso de outras estruturas, apenas não "estrague" a lista dada.
17. Dada uma **lista linear** contendo números inteiros. Construir uma função que insere uma nova informação V após a posição que possui a informação armazenada em X. Supor X e V valores lidos no *main* e passados na lista de parâmetros. Não usar estruturas auxiliares para a solução, tais como, vetor ou outra lista linear.
18. Dada uma **lista linear** contendo números inteiros. Construir uma função que insere após cada elemento, uma cópia dele. No final a lista deverá estar com

todos os valores duplicados. Não usar estruturas auxiliares para a solução, tais como, vetor ou outra lista linear.

19. Dada uma **lista linear** contendo números inteiros. Construir uma função que elimina o primeiro elemento da lista e coloca no seu lugar o último elemento da lista.
20. Dada uma **lista linear** contendo números inteiros. Construir uma função que elimina todos os elementos pares da lista.
21. Dada uma **lista linear** contendo em cada posição um caractere referente à uma palavra. Construir uma função que insere entre cada caractere da palavra, o caractere '*'.

22. Dada uma **lista linear** contendo números inteiros. Construir uma função que elimina o elemento seguinte à informação X, lido. Se X não existir na lista ou não existir o elemento seguinte ao X, retornar zero pelo *return*. Caso exista X e o seguinte, eliminar o seguinte e retornar 1 pelo *return*.
23. Dada uma **lista linear** contendo, **em cada nó um caractere** referente à uma frase. Construir uma função que elimina todos os elementos cujo caractere seja igual à vogal 'a'.

24. Polinômios

Seja o polinômio: $P(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1x^1 + a_0$,
onde, a_i : ***float*** e i: ***int*** - deverão ser armazenados numa lista.

```
typedef struct {
                    float coeficiente;
                    int expoente;
                  } termo;
typedef termo elemento_lista;
```

Supor: Lista P, Q, R; // nome das listas que deverão conter os polinômios

Construir as seguintes funções considerando: lista ordenada por expoente e sem coeficientes nulos:

a) Leitura: leitura e armazenamento de **um** polinômio numa lista.
void ***Leitura*** (Lista& S);

b) Soma: soma de dois polinômios gerando um terceiro. Não esquecer que os polinômios a serem somados estão armazenados em listas e o terceiro também deverá estar.
void ***Soma*** (Lista X, Lista Y, Lista& Z);

X, Y : polinômios de entrada e Z: polinômio de saída.

c) Mult: multiplicação de dois polinômios gerando um terceiro.
void ***Mult*** (Lista X, Lista Y, Lista& Z);

X, Y : polinômios de entrada e Z: polinômio de saída.