Lab_04

//----------

Prazo de entrega: 07/05/2025

Valor: 0

Não entrega: -5 valores

//----------

1. Crie um TAD Matriz, que deverá contar com as seguintes funções na sua interface matriz.h:

```
typedef struct matriz {
        int lin;
        int col;
        float  *v;  // implemente com array simples
} Matriz;

Matriz * mat_cria (int m, int n);
void mat_liberta (Matriz *mat);
float mat_acede_elemento (Matriz *mat, int i, int j);
void mat_atribui_elemento (Matriz *mat, int i, int j, float v);
int mat_linhas (Matriz *mat);
int mat_colunas (Matriz *mat);
int * mat_ordem_otima_multiplicacao (int *dimensoes);
```

   Exemplo de multiplicação de 3 matrizes (A1, A2 e A3) com retorno esperado da função mat_ordem_otima_multiplicacao. As matrizes têm a seguintes dimensões para este exemplo:
   A1: 5 linhas x 10 cols
   A2: 10 linhas x 3 cols
   A3: 3 linhas x 20 cols

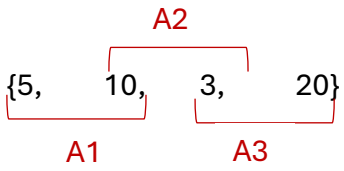
   Neste caso temos somente 2 possibilidades de multiplicação: (A1 x A2) x A3 ou A1 x (A2 x A3)

   - (A1 x A2) x A3
     - Total de 450 multiplicações escalares: (5x3x10)+(5x20x3)
   - A1 x (A2 x A3)

- Total de 1600 multiplicações escalares: (10x20x3)+(5x20x10)

Array de entrada da função: int dimensoes[] = {5,10,3,20} , pois

- A1 tem dimensões 5X10: {5,10,3,20}
- A2 tem dimensões 10X3: {5,10,3,20}
- A3 tem dimensões 3X20: {5,10,3,20}



A ordem de entrada das matrizes é sempre a mesma: A1, A2, A3. No retorno da função, temos a ordem em que as matrizes deverão ser multiplicadas, mantendo sempre a ordem de entrada como referência (A1, A2, A3).

A função mat_ordem_otima_multiplicacao deverá devolver:

int ordem_otima[] = {3,1,2}, onde:

- 3 indica que a matriz A1 deverá ser a terceira matriz a ser considerada na multiplicação
- 1 indica que a matriz A2 deverá ser a primeira matriz a ser considerada na multiplicação
- 2 indica que a matriz A3 deverá ser a segunda matriz a ser considerada na multiplicação

Ordem ótima da multiplicação das matrizes: A3 x (A2 x A1) é o que indica este retorno: {3,1,2}.

Este exercício deverá ser construído com modularização e deverá incluir programação dinâmica. Esperam-se os quatro seguintes ficheiros dentro do folder TAD_matriz:

matriz.h
matriz.c
main.c
Makefile

2. Você deseja arrumar as suas malas para uma viagem e precisa selecionar quais itens deverá levar em sua mala de mão, que tem capacidade máxima limitada. Cada item disponível tem um peso específico e um valor que representa a sua importância. O seu objetivo é maximizar o valor total dos itens carregados sem exceder a capacidade máxima da sua mala de mão.

Crie um TAD Mala, que deverá contar com as seguintes funções na sua interface mala.h:

```
typedef struct item {
        int id;
        float peso;
        float valor;
} Item;

typedef struct mala {
        float capacidade_max;
        Item *itens;
} Mala;

typedef struct mala_solucao {
        float valor_total;
        Item *itens;
} Mala_solucao;

Mala  * mala_cria (float capacidade_maxima);
void mala_liberta (Mala *m);
float mala_acede_capacidade_maxima (Mala *m);
Item * mala_acede_itens (Mala *m);
int mala_insere_item (Mala *m, Item *it);
int mala _remove_item (Mala *m, Item *it);
Mala_solucao * mala_solucao_otima (Mala *m, Item *it);
```

Este exercício deverá ser construído com modularização e deverá incluir programação dinâmica. Esperam-se os quatro seguintes ficheiros dentro do folder TAD_mala:

mala.h
mala.c
main.c

Makefile
Para a main.c, considere:

Capacidade máxima da mala = 8kg
Lista de itens:

id	peso (kg)	valor (€)
1	5	1000
2	3	800
3	3	800
4	2	400
5	4	750

O grupo deverá escrever um relatório de, no máximo, 7 páginas sobre estes experimentos. O relatório deverá conter: introdução, objetivo, resultados, discussão dos resultados e conclusão.

Na submissão, são esperados: relatório_grupoXX (pdf), TAD_matriz (zip) e TAD_mala (zip).