Miniteste 2 – Exemplo 2

1 – Escreva um programa que receba e processe os acessos de carros num parque de estacionamento. Um carro tem no máximo um acesso e cada acesso é definido por 3 valores inteiros com o seguinte formato: `numero entrada saída`.

O primeiro valor refere-se ao número do identificador do carro; os dois restantes valores referem-se aos instantes de tempo (em minutos) da entrada e saída do carro. A leitura termina quando não for possível ler os 3 valores. Considere que são sempre lidos 100 acessos.

O seu programa deve determinar qual o carro que permaneceu mais tempo no parque de estacionamento. Implemente e utilize no seu programa as seguintes funções:

- `int ler_acessos (int id[], int tempo[]);`

  Lê número do identificador para o vetor `id` e tempo correspondente para a mesma posição no vetor `tempo`. Retorna o número de acessos lidos.

- `int mais_tempo (int tempo[], int n);`

  Determina qual o carro que permaneceu mais tempo no parque. A posição onde está guardado o tempo de máximo é retornada pela função.

O seu programa pode ser testado com o ficheiro ***parque.txt*** [exemplo de utilização: ./prob1 < parque.txt]. Para esse ficheiro o resultado deverá ser:

```
Total de acessos: 100
Carro que permaneceu mais tempo: 18027 (216 minutos)
```

2 – Pretende-se criar um programa para efetuar a leitura das peças necessárias para a construção de um avião de aeromodelismo. Para o efeito pretende-se registar o nome da peça (uma palavra), a quantidade necessária e o custo unitário. Estude o ficheiro prob2.c do programa e complete-o de acordo como o que é pedido.

a) Desenvolva a função `ler_pecas` que lê um conjunto sucessivo de peças e retorna o número de peças lidas. A informação de cada peça é introduzida pelo utilizador. Garanta que não são introduzidas mais do que `N_PECAS` peças.

- `int ler_pecas(peca modelo[]);`

  Lê todas as peças do modelo para o vetor `modelo` e retorna número de peças armazenadas nesse vetor.

O seu programa pode ser testado com o ficheiro ***modelo.txt*** [exemplo de utilização: ./prob21 < modelo.txt]. Para esse ficheiro o resultado deverá ser:

```
*** Lista de pecas (19) ***
Balsa_36" - 10 - 48.04
...
Tube32" - 4 - 55.94
```

b) Implemente a função `lista_compras` que determina as peças a comprar, tendo em conta um limite de custo. Isto é, se o custo total (custo x quantidade) para uma determinada peça ultrapassa esse limite então não é inserida na lista.

```
peca* lista_compras(peca modelo[], int Nm, float limite, int *Nc);
```

Retorna um vetor contendo a lista de compras. O número de elementos nesse vetor é devolvido por referência no parâmetro `Nc`. Os parâmetros `modelo` e `Nm` são o vetor com as peças do modelo e o respetivo tamanho. O parâmetro `limite` indica o limite de custo por peça.

O seu programa pode ser testado com o ficheiro ***modelo.txt*** [exemplo de utilização: ./prob22 < modelo.txt]. Para esse ficheiro o resultado deverá ser:

```
Lista de compras (6) ***
Oleo_bracket - 2 - 10.44
...
Tube12" - 1 - 24.20
```

c) Implemente a função `guarda_lista` que grava num ficheiro a lista de compras a efetuar.

```
void guarda_lista(peca lista[], int n, char *nomeFicheiro);
```

Guarda no ficheiro com nome especificado por `nomeFicheiro` a lista de compras. Os parâmetros `lista` e `n` são o vetor com a lista de compras e o respetivo tamanho.

d) Admita que se pretende determinar a lista de compras tendo em conta o *stock* de peças já existente. Isto é, apenas as peças que não se encontram em *stock* devem ser compradas. Descreva sucintamente, e de forma clara, um algoritmo que lhe permita determinar essa lista de compras.