Miniteste 2 - Exemplo

1 – Um sistema de gestão de frotas de táxi encontra-se em desenvolvimento, e nesta fase pretende-se implementar uma forma de analisar os consumos. Escreva um programa que receba e guarde num vetor os quilómetros percorridos por um táxi entre sucessivos abastecimentos. O número de valores lidos é especificado pelo utilizador, até um máximo de 100. Considere que em cada abastecimento são sempre colocados 60 litros no depósito e que estes são gastos na totalidade até ao próximo abastecimento. Implemente e utilize no seu programa as seguintes funções:

```
int ler_quilometros(float *kms);
/* lê e guarda no vector kms as distâncias percorridas por um
táxi e retorna o número de valores efectivamente lidos */

float consumo_medio(float *kms, int n, int litros_dep);
/* calcula consumo medio em litros/100km tendo em conta as
distâncias percorridas (vector kms com tamanho n) e a capacidade
do depósito (em litros_dep)*/
```

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

```
Foram lidos 81 valores.
O consumo medio do táxi e' 7.55 l/100km.
```

2 – Desenvolva uma aplicação para gerir uma lista de produtos. Para cada produto pretende-se registar o nome (2 palavras), o preço, e a quantidade. Estude o ficheiro prob2.c do programa e complete-o de acordo como o que é pedido.

a) Desenvolva uma função que, garantindo que não são lidos mais do que NPRODUTOS, preencha o vetor loja com informação introduzida pelo utilizador; a função deve retornar o número de produtos lidos. A leitura deverá terminar quando não for possível ler todos os dados de um produto. Sugestão: verifique o valor de retorno da função scanf.

```
int ler_produtos(produto *loja);
```

O seu programa pode ser testado com o ficheiro loja.txt [exemplo de utilização: ./prob2 < loja.txt]. Para esse ficheiro o resultado deverá ser:

```
Lista de produtos carregada (54 produtos)
```

b) Desenvolva uma função que guarde a lista de produtos num ficheiro (nome do ficheiro indicado pelo parâmetro nomeFicheiro) e retorne o valor total dos produtos da loja. Utilize-a no programa apresentado.

```
float guarda_loja (produto *loja, int n, char *nomeFicheiro);
```

O seu programa pode ser testado com o ficheiro loja.txt [exemplo de utilização: ./prob2 < loja.txt]. Para esse ficheiro o resultado deverá ser:

```
Lista de produtos carregada (54 produtos)
Produtos guardados com o valor de: 7626.00
```

c) Considere que o utilizador pretende adicionar mais do que NPRODUTOS. Altere o seu programa de modo a que seja possível ao utilizador introduzir qualquer número de produtos. Sugestão: utilize gestão dinâmica da memória.

Comece por alterar o protótipo da função ler_produtos para:

```
produto* ler_produtos(int *n);
```

Não se esqueça de alterar no seu programa também as chamadas à função. O seu programa pode ser testado com o ficheiro loja2.txt [exemplo de utilização: ./prob2 < loja2.txt]. Para esse ficheiro o resultado deverá ser:

```
Lista de produtos carregada (108 produtos)
O valor dos produtos da loja e': 15253.76
```

d) Considere que o vetor de produtos (loja) está ordenado por ordem alfabética do nome de cada produto. Descreva um algoritmo eficiente que lhe permita procurar no vetor um determinado produto, sabendo o seu nome.