1 Considera uma loja que disponibiliza bebidas e que regista os pedidos dos clientes feitos ao longo do dia. As bebidas disponíveis são café a 0.65€ e leite a 0.50€. Escreve um programa que lê os pedidos de clientes e determina: o lucro total, quantas bebidas de cada tipo foram pedidas e qual a bebida preferida dos clientes.

Quando um cliente faz um pedido tem que indicar a bebida que pretende e a quantidade, sendo café indicado com a letra 'C' e leite com 'L'. Por exemplo C 2 (dois cafés), L 4 (quatro leites). Quando a loja fecha o empregado insere a letra 'F'. Considera que no máximo são lidos 100 pedidos.

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

```
Lucro total foi de 136.80 euros.
Foram pedidos 122 cafes e 115 leites.
A bebida preferida dos clientes e' cafe.
```

2 A NASA pretende manter o registo diário das horas de voo dos seus pilotos. Para tal necessita de uma aplicação para processar esses dados. O ficheiro **prob2.c** contém uma implementação incompleta da aplicação. Analisa com atenção antes de implementar as alíneas seguintes.

2.1 Implementa a função `ler_voos` que 1) lê a identificação do piloto (valor inteiro entre 1 e 60) e as horas que voou, e 2) guarda num vetor as horas, na posição correspondente ao número do piloto. A função recebe o vetor `horas`, que deverá ser preenchido, e retorna o número total de pilotos com voo registado; a função deverá também garantir que os limites do vetor não são ultrapassados.

```
int ler_voos(float horas[])
```

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

```
Numero total de pilotos com voo registado: 54
```

2.2 Implementa a função `processa_voos` que calcula a média de tempo de voo efetuado apenas pelos pilotos que efetuaram um voo (tempo > 0). A função devolve ainda os pilotos que têm tempo de voo abaixo da média.

```
float processa_voos(float horas[],int pilotos[], int *nPilotos)
```

A função recebe o vetor `horas`; retorna a média de horas e devolve as identificações dos pilotos abaixo da média no vetor `pilotos` e o respetivo tamanho em `nPilotos`.

O teu programa pode ser testado com o ficheiro ***voos.txt*** [exemplo de utilização: ./prob2 < voos.txt].

```
Media de horas de voo: 2.5
Pilotos com voos abaixo da media (26):
1 3 7 8 11 12 17 19 21 22 23 24 26 27 28 31 33 34 35 38 39 45 46 47 52 53
```

3 Para responder à necessidade de gerir os pórticos à volta da cidade do Porto, está a ser desenvolvido um programa capaz de processar a informação sobre os veículos que passam nesses pórticos. Cada passagem por um pórtico é identificada pelos seguintes valores: matrícula do veículo, pórtico (nome + valor), horas e minutos. Neste problema deves utilizar o ficheiro **prob3.c** que contém uma implementação incompleta. Analisa-o com atenção antes de implementar as alíneas seguintes.

3.1 Implementa a função `filtra_matricula` que devolve todas passagens de um veículo, identificado pela respetiva matrícula.

```
int filtra_matricula(passagem registos[], int nPassagens, char *matricula,
                     passagem resultado[])
```

A função recebe o vetor `registos` com todas as passagens, o respetivo tamanho `nPassagens` e a `matricula` do veículo; deverá preencher o vetor `resultado` com todas as passagens desse veículo e retornar o respetivo tamanho desse vetor. O vetor deverá estar por ordenado por ordem decrescente do valor do pórtico.

Depois de implementada a função, o programa deverá apresentar:

```
*** Passagens de 34-56-RI (8) ***
34-56-RI - Matosinhos_RTY, 2.3, 09:53
34-56-RI - Povoa_FGH, 1.8, 20:00
34-56-RI - Povoa_FGH, 1.8, 21:24
...
34-56-RI - Matosinhos_UIO, 0.9, 17:29
34-56-RI - Porto_ZXC, 0.9, 07:22
34-56-RI - Porto_FGH, 0.5, 09:53
```

3.2 Implementa a função `guardaTotais` que guarda num ficheiro de texto os valores totais de cada pórtico, ou seja, a soma dos valores registados para as passagens em cada pórtico.

```
float guardaTotais(passagem registos[], int nPassagens, char *nomeFicheiro)
```

A função recebe o vetor `registos` com todas as passagens, o respetivo tamanho `nPassagens` e o nome do ficheiro onde são guardados os totais em `nomeFicheiro`. A função retorna a soma dos totais e 0 em caso de erro a abrir o ficheiro.

Depois de implementada a função, o programa deverá apresentar:

```
*** A gravar valores totais de cada portico para o ficheiro totais.txt ***
Total de 71.5 euros gravado com sucesso
```