# Programação - Exame da época especial

Engª Informática; Engª Informática – Pós-laboral; Engª Informática - Curso Europeu

Duração: 2h30m 06/09/2017

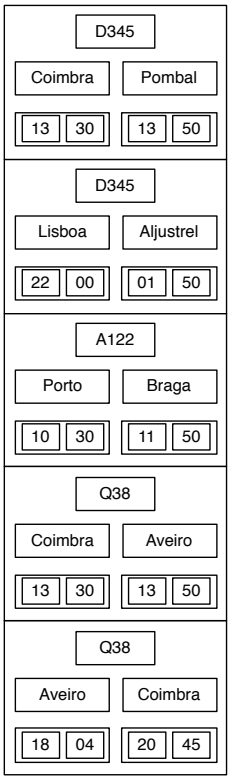
**Atenção**: É obrigatório apresentar uma estratégia genérica para cada um dos exercícios.

**1.** Considere as seguintes definições:

```
struct hora{
    int h, m;
};

typedef struct{
    char id[15];
    char in[50], out[50];  // locais de entrada/saída
    struct hora inH, outH; // horas de entrada/saída
}util;

typedef struct{
    char id[15];
    int valorTotal;
}resumo;
```




As utilizações do sistema Via Verde durante um determinado período estão armazenas num ficheiro binário com estruturas do tipo *util*.
Cada estrutura refere-se a uma utilização completa do sistema contendo a identificação do utilizador, os locais de entrada e saída e as respetivas horas de entrada e saída. Ao responder a esta questão pode assumir que nenhuma utilização individual ultrapassa as 12 horas e que os locais de entrada e saída são constituídos apenas por uma palavra. No ficheiro exemplificado ao lado existem 5 utilizações de 3 utilizadores diferentes.

**a)** Desenvolva uma função em C que elimine do ficheiro binário todas as utilizações inferiores a uma hora. A função recebe como parâmetro o nome do ficheiro. Deve atualizar o ficheiro da forma pretendida e devolver o número de estruturas eliminadas.

**b)** Um ficheiro de texto armazena os valores a pagar entre todos os pares entrada/saída que podem ser usados na autoestrada. Em cada linha deste ficheiro surge a identificação de 2 locais seguidos pelo valor inteiro a pagar, como se pode ver no exemplo ao lado. Todos os veículos pagam o mesmo e a direção seguida é indiferente (Aveiro-Coimbra tem o mesmo custo de Coimbra-Aveiro)

```
Lisboa  Coimbra  12
Aveiro  Coimbra  5
...
```

Desenvolva uma função em C que crie um vetor dinâmico do tipo *resumo* com informação sobre o valor a pagar por cada utilizador no período registado no ficheiro binário. Cada estrutura pertencente ao vetor deve armazenar os dados de um utilizador, sendo o valor total a pagar contabilizado pela função com ajuda da informação existente no ficheiro de texto. Considerando o exemplo em cima, deveria ser criado um vetor dinâmico com 3 estruturas. O cabeçalho da função é o seguinte:

```
resumo* criaVetor(char* nomeBin, char* nomeTXT, int* total);
```

Devolve o endereço inicial do vetor dinâmico criado dentro da função (NULL caso exista algum erro) e recebe como parâmetros os nomes dos ficheiros e o endereço de uma variável inteira onde deve ser colocada a dimensão do vetor.

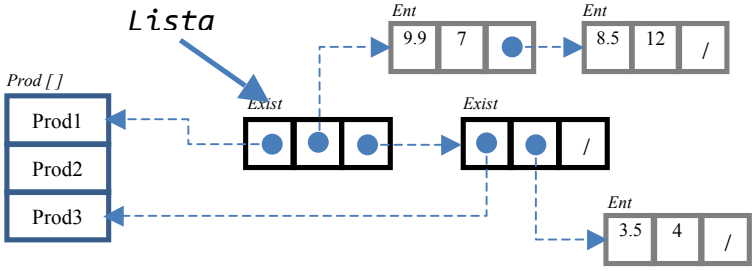
**2.** Considere as seguintes definições que permitem criar uma estrutura dinâmica para armazenar produtos de um supermercado:

```
typedef struct produto Prod, *pProd;
typedef struct entrada Ent, *pEnt;
typedef struct existencia Exist, *pExist;

struct produto {          // representa um produto
  int id;                 // identificador único do produto
  char designacao[100];
  char familia[30];       // fruta, detergente, lacticínios, ...
};

struct entrada {            // representa uma entrada de stock
  int qt;                   // qtd. que deu entrada
  float preco_compra;       // preço unitário relativo a esta entrada
  pEnt prox;
};

struct existencia { // representa o conjunto de abastecimentos de um produto
  pProd p;          // produto a que se refere esta estrutura
  pEnt entradas;    //lista com as diversas entradas deste produto
  pExist prox;
};
```




A estrutura dinâmica é constituída por um vetor dinâmico de *Prod* onde se encontram armazenados todos os produtos. Existe igualmente uma lista ligada constituída por nós do tipo *Exist* com as existências dos produtos. Cada nó da lista principal referencia um produto específico (através do ponteiro *p*) e tem pendurada uma lista de entradas em stock desse produto. Cada entrada em stock é caracterizada por uma quantidade (número de unidades adquiridas) e pelo preço unitário.

**a)** Desenvolva uma função em C que escreva na consola o *id* e *designação* de todos os produtos que tenham uma quantidade em stock superior a um determinado limite. A função recebe como parâmetros o ponteiro para a lista ligada de existências de todos os produtos e o limite mínimo em stock a considerar.

**b)** Desenvolva uma função em C que adicione um novo produto à estrutura dinâmica, incluindo várias entradas em stock. A informação do novo produto a adicionar está armazenada num ficheiro de texto com o seguinte formato: na primeira linha está o *id*, *designação* e *família* do novo produto (uma palavra cada). A seguir estão várias linhas, cada uma delas contendo uma entrada em stock do produto. Pode assumir que existe pelo menos uma entrada em stock.

| | | |
|---|---|---|
| P123 | Simplex | Detergente |
| 12 | 3.6 | |
| 5 | 2.9 | |

A função deve criar espaço no vetor para o novo produto, atualizar a lista ligada de existências e criar a lista ligada de entradas em stock desse produto. O cabeçalho da função é o seguinte:

```
pProd func2B(pProd v, int * totProd, pExist* lista);
```

Recebe como parâmetros um ponteiro para o início do vetor, o endereço de uma variável inteira onde está armazenada a dimensão do vetor e o endereço do ponteiro para o início da lista ligada. Devolve um ponteiro para o início do vetor de produtos atualizado. O número de elementos no vetor e o ponteiro para o início da lista devem ser atualizados diretamente através dos parâmetros.