Programação - Exame Normal

Eng.ª Informática; Eng.ª Informática – Pós-laboral; Eng.ª Informática - Curso Europeu

Duração: 2h30m 11/06/2019

1. Um **ficheiro de texto** contém informação sobre as classificações obtidas numa **corrida com 10 Km**.
Os dados mostram a classificação que se verificava no final de cada um dos km, com indicação da posição de cada atleta e o respetivo tempo acumulado. Pode assumir que não existem desistências ao longo da prova. No ficheiro ao lado pode ver-se um exemplo para uma prova em que participaram 3 pessoas. A primeira linha contém o número de atletas que participaram. Depois disso surgem várias linhas com a classificação no final de cada um dos km. O ficheiro de texto não tem linhas em branco no meio da informação. A informação relativa a cada km tem uma linha de cabeçalho com indicação *Km x* (x é um inteiro indicando o km), seguida de N linhas (1 para cada participante), cada uma com o formato indicado no exemplo: posição, ponto, espaço, nome, dois pontos, espaço e tempo acumulado (inteiro).
Considerando o exemplo ao lado, no final do 1º km o Pedro Mendes está em primeiro lugar com 45 segundos. No final do 2º km, a Ana Alves está na frente com um tempo total de 90 segundos, significando que demorou 44 segundos neste 2º km. O vencedor da corrida foi o Pedro Mendes com um tempo total de 489 segundos.

Corrida_10KM.txt

```
Atletas: 3
Km 1
1. Pedro Mendes: 45
2. Ana Alves: 46
3. Zulmira Pires: 57
Km 2
1. Ana Alves: 90
2. Pedro Mendes: 101
3. Zulmira Pires: 103
Km 3
1. Ana Alves: 150
2. Pedro Mendes: 151
3. Zulmira Pires: 158
….
Km 10
1. Pedro Mendes: 489
2. Zulmira Pires: 510
3. Ana Alves: 570
```

a [4 vals.]. Escreva uma função em C que calcule quanto tempo decorreu entre a chegada do primeiro e do último concorrente. A função recebe o nome do ficheiro como parâmetro e devolve o valor contabilizado. Caso exista algum erro, a função devolve -1.

b [5 vals.]. Escreva uma função em C que crie um vetor dinâmico com um resumo da informação relativa à participação de cada atleta numa corrida. Cada atleta deve ter a sua informação armazenada numa estrutura do tipo *struct atleta*.

```
struct atleta{
      char nome[100];
      int tAcumulado[10];
      int kmRapido;
};
```

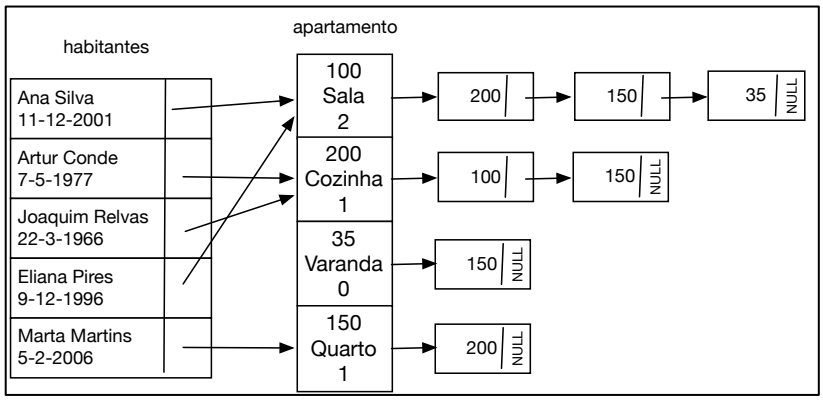
Cada estrutura permite armazenar o nome, o tempo acumulado no final de cada km e o tempo registado no km mais rápido. A ordem pela qual as estruturas dos atletas surgem no vetor dinâmico é irrelevante e pode assumir que os nomes são únicos. A função recebe como como parâmetros o nome do ficheiro de texto e o endereço de uma variável inteira onde deve ser colocada a dimensão do vetor dinâmico. Devolve o endereço do vetor criado pela função, ou NULL, em caso de erro.

2 [2 vals.]. Uma árvore binária constituída por nós do tipo *no* está **ordenada de forma crescente** pelos valores do campo *num*. Pode assumir que não existem valores repetidos na árvore.

```
typedef struct valor no, *pno;
struct valor{
      int num;
      pno dir, esq;
};
```

Escreva uma **função recursiva** em C que escreva na consola todos os valores armazenados em folhas da árvore. Os valores devem ser escritos por **ordem decrescente**.

3. Uma estrutura dinâmica armazena informação sobre as várias divisões de um apartamento e sobre as pessoas que aí se encontram. As estruturas do lado direito são utilizadas na implementação.




```
typedef struct divisao div, *pdiv;
typedef struct porta no, *pno;

struct divisao{
    int id;
    char nome[100];
    int n_pessoas;
    pno lista;
};

struct porta{
    int id;
    pno prox;
};

struct pessoa{
    char nome[100];
    struct{
       int d,m,a;
    }nasc;
    pdiv local;
};
```

Na figura pode ver um exemplo de uma estrutura dinâmica deste tipo. Existe um vetor dinâmico *apartamento* contendo estruturas do tipo *div*, uma para cada divisão do apartamento. Cada divisão tem um id único, um nome e um contador das pessoas que aí se encontram. De cada elemento do vetor sai uma lista ligada do tipo *no* que contém as divisões que é possível atingir diretamente: por exemplo, a partir da cozinha é possível passar diretamente para as divisões com id 100 e 150. Neste apartamento há portas que têm um sentido e outras portas que têm 2 sentidos. Como se pode ver na figura, é possível ir da sala para a cozinha e vice-versa. Por outro lado, é possível ir da varanda para o quarto, mas não o inverso (não há ligação no sentido quarto, varanda).
Existe um outro vetor dinâmico *habitantes*, com estruturas do tipo *struct pessoa*, contendo informação sobre as pessoas que estão em casa. Cada pessoa tem um nome, data de nascimento e um ponteiro para a divisão em que encontra, *i.e.,* para a respetiva estrutura no vetor de divisões.

a [4 vals]. A estrutura dinâmica pode estar corrompida e não se sabe se os contadores do número de pessoas nas divisões estão corretos. No exemplo da figura em cima pode confirmar que o contador da cozinha está errado, uma vez que se encontram duas pessoas nesta divisão. Escreva uma função em C que faça esta verificação e que corrija os contadores que eventualmente se encontrem incorretos. A função recebe os endereços e dimensões dos 2 vetores dinâmicos (divisões e pessoas). Devolve como resultado um inteiro indicando quantos contadores foi necessário atualizar.

b [5 vals.]. Escreva uma função em C que elimine uma divisão da estrutura dinâmica. Todo o espaço ocupado por esta divisão (estrutura no vetor e lista de ligações) deve ser libertado. As pessoas que eventualmente se encontrem neste local devem ser movidas para uma outra divisão da casa, para a qual exista uma ligação direta. Pode assumir que ainda vão restar outras divisões para onde as pessoas se podem deslocar. Tenha em consideração que poderá ter que atualizar as listas de outras divisões da casa, uma vez que estas poderão conter ligações para a que vai ser eliminada. A função tem o seguinte protótipo:

```
pdiv elimina(pdiv d, int *totD, int id, struct pessoa* hab, int totP);
```

Recebe como parâmetros o endereço do vetor de divisões, o endereço de uma variável contendo o número total de divisões antes da eliminação, o identificador da divisão a eliminar e o endereço e dimensão do vetor de habitantes. Devolve o endereço do vetor de divisões depois da atualização.