Ficha 7

Programação Imperativa

Listas

Use, se achar necessário, o projecto `https://codeboard.io/projects/238187` para responder aos problemas propostos.
Considere que se pretende armazenar quantas vezes uma determinada palavra aparece num texto.
Para isso definiu-se o seguinte tipo (listas ligadas em que cada elemento armazena uma palavra e o número de vezes que essa palavra ocorre).

```
typedef struct celula {
    char *palavra;
    int ocorr;
    struct celula * prox;
} * Palavras;
```

Assume-se que a lista não tem palavras repetidas.
Apresente definições para as seguintes funções:

1. `void libertaLista (Palavras)` que liberta todo o espaço (da *heap*) ocupado pela lista de palavras.

2. `Lista quantasP (Palavras l)` que calcula quantas palavras (diferentes) existem armazenadas.

3. `void listaPal (Palavras l)` que escreve no ecran, uma por linha, todas as palavras armazenadas, bem como o número de ocorrências.

4. `char * ultima (Palavras l)` que determina qual a última palavra da lista.

5. `Palavras acrescentaInicio (Palavras l, char *p)` que acrescenta uma palavra no início da lista (com `1` como número de ocorrências).

6. `Palavras acrescentaFim (Palavras l, char *p)` que acrescenta uma palavra no fim da lista (com `1` como número de ocorrências).

7. `Palavras acrescenta (Palavras l, char *p)` que regista mais uma ocorrência da palavra `p`. Se a palavra já existir, o número de ocorrências deve ser incrementado. No outro caso deve ser inserida uma nova célula.

   A inserção de uma nova célula pode ser feita no início da lista ou ordenadamente, assumindo que a lista está armazenada por ordem alfabética.

8. `struct celula * maisFreq (Palavras l)` que calcula a célula correspondente à palavra mais frequente.