Programação em C - Exercícios de revisão

1.1 Escreva uma função `int contar_maiores(int vec[], int size, int val)` cujos argumento são uma variável indexada `vec` com tamanho `size` e um valor `val` e cujo resultado deve ser a contagem do número de elementos de `vec` que são estritamente maiores do que `val`.

1.2 Escreva uma função `int palindromo(char str[])` que testa se uma cadeia de carateres é um palíndromo, isto é, se tem a mesma sequência de carateres da esquerda para a direita e vice-versa.

1.3 Escreva uma função `int decimal(char str[])` que converte uma cadeia de carateres com algarismos de 0 a 9 no valor inteiro decimal correspondente. Por exemplo: `decimal("1234")` deve dar retornar o inteiro 1234.

1.4 Escreva uma função `int repetidos(int vec[], unsigned size)` que testa se há (pelo menos) dois valores iguais no vector `vec` com tamanho `size`; o resultado deve ser 1 em caso afirmativo e 0 em caso negativo. Exemplos:

```
int a[5] = { 2, -1, 0, 2, -1 };
int b[5] = { 3, 4, 1, 2, -1 };
printf("%d\n", repetidos(a, 5)); // imprime 1
printf("%d\n", repetidos(b, 5)); // imprime 0
```

Tenha atenção que a sua função não modifique os elementos do vector passado como argumento.

1.5 Escreva uma definição da função

```
void range(int vec[], unsigned size, int inicio, int incr)
```

que inicializa elementos de um vector `vec` com `size` valores inteiros `inicio`, `incio+incr`, `incio+2*incr`, etc. seguindo uma progressão aritmética. Exemplo:

```
int a[5];
range(a, 5, 3, 2); /* a[] passa a conter { 3, 5, 7, 9, 11 } */
```

1.6 Escreva uma função `int filtrar_positivos(int vec[], int size)` que remove os valores não positivos (isto é, negativos ou zero) de um vector `vec` com tamanho `size`.

A função deve modificar a variável indexada dada de forma a que os valores positivos fiquem num segmento inicial do vetor. O resultado deve ser o número de valores positivos (i.e. o comprimento do segmento final).

1.7 Implemente uma função `void ordenar(char str[])` que ordena os carateres numa cadeia pelos seus códigos. Por exemplo: se `str = "ALGORITMO"` então após execução devemos ter `str = "AGILMOORT"`.

Sugestão: cadeias de carateres em linguagem C são variáveis indexadas. Pode usar um dos algoritmos de ordenação da sua preferência (e.g., seleção, inserção, "quicksort"). Pode determinar o número de carateres da cadeia usando `strlen`.

1.8 Escreva uma função `int anagramas(char str1[], char str2[])` que determina se duas cadeias de carateres são *anagramas*, isto é, se se escrevem com os mesmos carateres. O resultado deve ser 1 em caso afirmativo e 0 caso contrário. Por exemplo, "deposit" e "topside" são anagramas:

```
char str1[] = "deposit";
char str2[] = "topside";
int r = anagramas(str1,str2); // resultado 1
```

Sugestão: use uma função auxiliar como no exercício anterior para ordenar ambas as cadeias; as cadeias originais são anagramas se e só se as cadeias ordenadas são *exatamente iguais* carater-a-carater.

1.9 Um *quadrado mágico* é uma matriz quadrada de números inteiros tal que todas as linhas, colunas e diagonais somam o mesmo valor. No exemplo seguinte cada linha, coluna e diagonal soma o mesmo valor (15 neste caso):

$$\begin{bmatrix} 2 & 7 & 6 \\ 9 & 5 & 1 \\ 4 & 3 & 8 \end{bmatrix}$$

Escreva uma função `int magico(int a[20][20], int n)` que testa se uma matriz é um quadrado mágico. A matriz é representada por uma variável indexada `a` com dimensão declarada 20×20; o argumento `n` indica qual sub-matriz a considerar: por exemplo, se `n = 3` devemos testar se a sub-matriz de 3×3 é quadrado mágico (e ignorar o resto da matriz). O resultado deve ser um inteiro: 1 se é quadrado mágico e 0 caso contrário.