Programação em C - Exercícios sobre passagem de argumentos pela linha de comandos e alocação dinâmica de memória

Ligações úteis:

- Slides de *Visão geral da linguagem C* disponibilizados no moodle;
- Command-Line Arguments in C por Jacob Sorber;
- Programming and Data Structures Videos (Neso Academy) - Alocação dinâmica de memória - Video 20 a 23

---

1. Escreva um programa **ccaract** que imprima o número de caracteres de cada parâmetro passado na linha de comandos, bem como o número total de caracteres. Escreva duas versões, uma que utilize a função `strlen()`, e outra que não a utilize.

```c
#include <stdio.h>

int main(int argc, char* argv[]){
  /*
   *  argc: número de argumentos da linha de comandos (incluindo o nome do executável)
   *  argv[]: vector com os argumentos (cadeias de caracteres) passados na linha de comandos:
   *  argv[0] - corresponde ao nome do executável
   *  argv[1] - corresponde ao primeiro argumento
   *  ...
   *  argv[argc-1] - corresponde ao último argumento
   *  argv[argc] - tem valor NULL
   */

  // completar programa...

    return 0;
}
```

Exemplo de execução:

```
$ ./ccaract ola 4 23 fim
ola -> 3
4 -> 1
23 -> 2
fim -> 3
total = 9
```

2. Escreva um programa que calcule e imprima a soma de uma lista de números passados como parâmetros na linha de comando.

   Exemplo:

```
$ ./soma 4 5 2 8 20
39
```

3. O seguinte fragmento de código tem por objetivo ler da entrada padrão duas sequências de inteiros de tamanho **n** (um inteiro por linha), e proceder à soma desses arrays guardando o resultado num novo array. Apesar do programa ainda não estar completo, podemos desde já identificar que apresenta **um problema muito grave**:

```c
#include <stdio.h>
#include <stdlib.h>

int* readarray(int n){
    int i;
    int v[n];
    for (i=0; i<n; i++){
        scanf("%d", v+i);
    }
    return v;
}

int* somaarrays(int *a, int *b, int n) {
    /* A completar:
      deverá retornar um novo array
      com a soma dos arrays arrays a e b,
      elemento a elemento
    */
}
void printarray(int *v, int n){
    // A completar:
    // deverá imprimir os elementos de v
}

int main(){
    int n;
```

```
    int *va, *vb, *vr;
    scanf("%d\n", &n);
    va = readarray(n);
    vb = readarray(n);
    vr = somaarrays(va, vb, n);
    printarray(va, n);
    printarray(vb, n);
    printarray(vr, n);
    return 0;
}
```

- Qual é esse problema?
- Complete o código do programa, corrigindo o problema identificado. Compile e teste o programa.

4. *Espaços!*
   - Implemente uma função com o protótipo `void esp(char *s);` que troca os caracteres de **s** que não são letras nem dígitos por espaços. Por exemplo, `"a, Ah Ah!"` fica transformada em `"a  Ah Ah "`.
   - Implemente uma função com o protótipo `char* esp1(char *s);` que retorna uma "string" resultante da troca de todos caracteres de **s** que não são letras nem dígitos por espaços (**s** não é alterada). O espaço para o resultado deve ser obtido pela função `malloc()`.
   - Escreva um pequeno programa para testar as funções acima pedidas.

5. Processamento de *Strings...*

Para a implementação das seguintes funções, consulte o manual da funções correspondentes da biblioteca *sting.h*.

- Implemente uma função `char* my_strncat(char *dest, char *src, int n)`, que acrescenta (copia), no máximo **n** caracteres da string **src** ao fim da string **dest**, retornando um apontador para a string resultante **dest**. Que cuidados é necessário ter na implementação e utilização desta função?
- Implemente uma função `char* my_strdup(char *s)`, que retorna uma nova string corresponde a uma cópia da string **s**.
- Escreva um programa para testar as funções acima pedidas.

6. Escreva um programa que leia uma lista de inteiros (o primeiro inteiro indica o tamanho da lista), e que escreva os inteiros da lista que aparecem **repetidos pelo menos uma vez** na lista dada, pela ordem da primeira vez que ocorreram no input.

   Exemplo de possível execução do programa:

```
$ ./repetidos
6
1
2
1
10
2
1

1
2
```

As duas últimas linhas constituem o resultado. Para resolver o exercício, deverá guardar num vector `a[]`, **alocado dinamicamente**, o conjunto dos inteiros que já foram lidos pelo menos uma vez. Note-se que é necessário distinguir o caso de o inteiro ter ocorrido uma vez, do de ter ocorrido duas ou mais vezes. Assim, sugere-se que cada elemento de `a[]` seja uma estrutura como indicado:

```
struct elemento{
    int valor; // um inteiro lido
    int ocorrencias; // numero de vezes que esse inteiro foi lido
};
typedef struct elemento ELEMENTOS;

ELEMENTOS *a = NULL;
```