PROGRAMAÇÃO 2 – LISTA 1

Prof. Thiago Cavalcante

1. Escreva um programa que leia do teclado **20 números inteiros entre 10 e 100** (faça essa verificação e informe o usuário!) e imprima na tela todos os valores **únicos** que foram lidos. Assuma que pode ocorrer o pior caso, onde todos os 20 números são únicos. Você pode resolver de duas formas:

   (a) na tela se o número é único ou não, no momento que ele é lido;

   (b) todos os números únicos em um array e imprimindo-os ao final da leitura dos 20 números.

   Em qualquer um dos casos, utilize o **menor array possível**. *(**dica:** as verificações de intervalos de números podem ser feitas com um laço **while**, que impede a continuação do programa até que o número lido no teclado esteja no intervalo desejado.)*

2. Escreva uma função **recursiva** chamada **minimo_recursivo** que recebe um array de inteiros e seu tamanho como argumentos e retorna o menor elemento do array. *(**dica:** você pode "reduzir" o tamanho de um array em uma função recursiva de duas formas: 1) passando a referência para um elemento posterior e diminuindo o tamanho de acordo: **array, tamanho -> &array[1], tamanho - 1**; 2) usando dois valores, **low** e **high**, para representar os índices mínimo e máximo a serem considerados no array: **array, low, high -> array, low + 1, high**)*

3. Escreva uma função **recursiva** chamada **testa_palindromo** que recebe uma string e seu tamanho como argumentos e retorna 1 se essa string for um palíndromo, e 0 se não for. Assuma que a string é uma palavra simples, sem acentos, pontuação ou espaços.

4. Escreva uma função **recursiva** chamada **string_invertida** que receba uma string como argumento, imprima-a de trás para a frente e não retorne nada. A função deverá encerrar o processamento e retornar quando o caractere de finalização da string '\0' for encontrado.

5. Escreva um programa que leia do teclado uma data no formato "12/10/2010" (não precisa fazer a verificação, assuma que a entrada sempre está nesse formato) e imprima na tela a mesma data, no formato "12 de outubro de 2010". *(**dica:** você pode guardar as strings com os nomes de cada mês em um array de strings **char *array_de_strings[tamanho] = {"string1", "string2", ...}**)*

6. Escreva um programa que leia do teclado um número inteiro entre **1 e 99** (faça a verificação) e imprima na tela o número escrito **por extenso**.

7. Escreva um programa que leia uma frase em português e codifique-a em código Morse. Assuma que a entrada contém apenas letras maiúsculas e espaços para separar as palavras (não contém acentos nem números). Quando imprimir na tela, use um espaço entre cada letra do código Morse e três espaços entre cada palavra em código Morse.

8. Dadas a estrutura e declarações abaixo,

```
struct cliente {
  char nome[15];
  char sobrenome[15];
  int numCliente;

  struct {
    char telefone[11];
    char endereço[50];
    char cidade[25];
    char estado[2];
    char cep[8];
  } pessoal;
} regCliente, *ptrCliente;

ptrCliente = &regCliente;
```

   escreva as expressões para acessar cada campo pedido a seguir:

   (a) Campo **sobrenome** da estrutura **regCliente**.

   (b) Campo **sobrenome** da estrutura apontada por **ptrCliente**.

   (c) Campo **nome** da estrutura **regCliente**.

   (d) Campo **nome** da estrutura apontada por **ptrCliente**.

   (e) Campo **numCliente** da estrutura **regCliente**.

   (f) Campo **numCliente** da estrutura apontada por **ptrCliente**.

   (g) Campo **telefone** do campo **pessoal** da estrutura **regCliente**.

   (h) Campo **telefone** do campo **pessoal** da estrutura apontada por **ptrCliente**.

   (i) Campo **endereço** do campo **pessoal** da estrutura **regCliente**.

   (j) Campo **endereço** do campo **pessoal** da estrutura apontada por **ptrCliente**.

   (k) Campo **cidade** do campo **pessoal** da estrutura **regCliente**.

(l) Campo **cidade** do campo **pessoal** da estrutura apontada por **ptrCliente**.

(m) Campo **estado** do campo **pessoal** da estrutura **regCliente**.

(n) Campo **estado** do campo **pessoal** da estrutura apontada por **ptrCliente**.

(o) Campo **cep** do campo **pessoal** da estrutura **regCliente**.

(p) Campo **cep** do campo **pessoal** da estrutura apontada por **ptrCliente**.

9. Escreva uma função que imprima na tela um retângulo sólido cujos lados são lidos nos parâmetros inteiros **lado_hor** e **lado_ver**. O retângulo deve ser formado com qualquer caractere lido no parâmetro **preenchimento**. Por exemplo, se **lado_hor** é 7, **lado_ver** é 5 e **preenchimento** é 'X', a função deverá imprimir

```
XXXXXXX
XXXXXXX
XXXXXXX
XXXXXXX
XXXXXXX
```

10. Escreva uma função que leia um número inteiro do teclado e retorne o número com seus dígitos invertidos. Por exemplo, dado o número 5234, a função deverá retornar 4325.