**Estrutura de Dados – Lista de Exercícios de Funções, Ponteiros e Tipos de Variáveis**

**P.S.: Códigos das questões devem ser incluidos (.c)**

1) Seja o trecho de código abaixo, quais serão os valores de x, y e *p no comando printf?

```c
int x, y, *p;
y = 0;
p = &y;
x = *p;
x = 4;
(*p)++;
--x;
(*p) += x;
printf("x=%d  y=%d *p=%d", x, y, *p);
```

2) Seja o trecho de código abaixo, que valor de c será impresso no comando printf?

```c
int a=5, b=12, c;
int *p;
int *q;
p = &a;
q = &b;
c = *p + *q;
printf("c = %d", c);
```

3) Seja o trecho de código abaixo, o uso do ponteiro está correto? Se sim, o que está sendo passado como parâmetros para a função "**swapValues**"?

```c
int main() {
    double a = 5.5;
    double b = 3.0;
    swapValues(&a, &b);
    return 0;
}

void swapValues(double *x, double *y) {
    double temp = *x;
    *x = *y;
    *y = temp;
}
```

4) Após a execução do Código abaixo, qual é o valor da variável **"num"?**

```c
int main() {
    int num = 42;
    int *ptr1 = &num;
    int *ptr2 = ptr1;
    *ptr2 = 99;
    return 0;
}
```

5) Considerando o código abaixo, indique os valores de x, y e *p bem como a sua pilha de execução:

```c
void main() {
    int x, y, *p; y = 0;
    p = &y;
    x = *p;
    x = 4;
    (*p)++;
    --x;
    (*p) += x;
}
```

6) Observe o seguinte trecho de Código e indique o valor de cada variável em cada linha:

```c
07.     a = 4;
08.     b = 3;
09.     p1 = &a;
10.     p2 = p1;
11.     *p2 = *p1 + 3;
12.     b = b * (*p1);
13.     (*p2)++;
14.     p1 = &b;
```

7) Escreva um programa que contenha duas variaveis inteiras. Leia essas variáveis do teclado. Em seguida, compare seus endereços e exiba o conteudo do maior endereço.

8) Elaborar um programa que leia dois valores inteiros (A e B). Em seguida faça uma função que retorne a soma do dobro dos dois numeros lidos. A função deverá armazenar o dobro de A na propria variável A e o dobro de B na própria variável B.

9) Qual é a saída do programa abaixo e por que?

```c
#include <stdio.h>

void increment() {
    static int count = 0;
    count++;
    printf("Count: %d\n", count);
}

int main() {
    increment();
    increment();
    return 0;
}
```

10) Qual é a diferença entre uma variável global e uma variável local em C?

a) Variáveis globais não podem ser modificadas após a declaração.
b) Variáveis locais não têm um valor inicial definido.
c) Variáveis locais só podem ser usadas dentro de funções.
d) Variáveis globais podem ser acessadas e modificadas por qualquer função no programa.

11. Crie uma `struct` chamada **Pessoa**, que tenha os seguintes campos:

- `nome` (string de até 50 caracteres)
- `idade` (inteiro)
- `altura` (float)

No programa principal:

1. Declare uma variável do tipo `Pessoa`.
2. Solicite ao usuário que insira os valores para os campos.
3. Exiba os dados inseridos na tela.

12. Crie uma `struct` chamada **Aluno**, que contenha:

- `nome` (string de até 50 caracteres)
- `matricula` (inteiro)
- `nota` (float)

O programa deve:

1. Criar um array para armazenar **3 alunos**.
2. Pedir ao usuário para inserir os dados de cada aluno.
3. Criar uma função chamada `exibirAlunos()` que recebe o array e imprime as informações dos alunos formatadas.

13. Crie uma `struct` chamada **Carro**, que contenha:

- `marca` (string de até 30 caracteres)
- `ano` (inteiro)
- `preco` (float)

O programa deve:

1. Criar uma variável do tipo `Carro`.
2. Criar um ponteiro para essa struct.
3. Pedir ao usuário para preencher os dados.
4. Exibir as informações acessando os valores pelo ponteiro.

14. Crie um programa que utilize **structs aninhadas** para armazenar informações de um endereço dentro de uma estrutura de Pessoa.

1. Crie uma `struct` chamada **Endereco**, contendo:
    - `rua` (string de até 50 caracteres)
    - `numero` (inteiro)
    - `cidade` (string de até 30 caracteres)
2. Crie uma `struct` chamada **Pessoa**, contendo:
    - `nome` (string de até 50 caracteres)

    - `idade` (inteiro)
    - `endereco` (struct Endereco)
3. Peça ao usuário para preencher os dados de **uma pessoa** e seu endereço.

15. Crie uma `struct` chamada **Produto**, que contenha:

- `nome` (string de até 50 caracteres)
- `preco` (float)
- `quantidade` (inteiro)

O programa deve:

1. Criar **uma função** chamada `calcularTotal()` que recebe um **Produto** como parâmetro e retorna o valor total (`preco * quantidade`).
2. No `main()`, pedir ao usuário para inserir os dados de um produto.
3. Exibir o total calculado chamando a função.