Exercícios – Ponteiros em C (sem gabarito)

Exercícios – Ponteiros em C
(2 fáceis, 6 médios, 2 difíceis)

FÁCEIS
1) (Conceitual) Considere:

```
int x = 10;
int *p = &x;
printf("%d\n", *p);
printf("%p\n", (void*)p);
printf("%p\n", (void*)&x;);
```

a) O que será impresso na primeira linha?
b) Qual a relação entre os dois endereços impressos nas últimas linhas?
c) Em uma frase, defina o que é *p e o que é &x.;

2) (Função simples) Implemente a função void zera(int *n) que recebe o endereço de um inteiro e altera
seu conteúdo para zero. Mostre um main mínimo que prova o funcionamento com uma variável int.

MÉDIOS
3) (Troca) Implemente a função void troca(int *a, int *b) que troca os valores de duas variáveis inteiras,
e demonstre seu uso em main com x=3 e y=5.

4) (Somatório com ponteiros) Dado int v[5] = {1,2,3,4,5}; e int *p = v;
a) Escreva um laço que some todos os elementos usando apenas aritmética de ponteiros (sem índices).
b) Qual é o valor de *(p+3)? Justifique.

5) (scanf e &) Explique e corrija o erro no código abaixo para ler um inteiro do teclado:

```
int num;
scanf("%d", num);
```

Escreva a versão correta e explique por que o operador & é necessário.

6) (Preenchimento por referência) Escreva a função void preenche(int *v, int n) que armazena v[i] = 2*i
para i = 0..n-1 utilizando apenas aritmética de ponteiros. Demonstre no main imprimindo o vetor.

7) (malloc, verificação e free) Complete o programa para alocar dinamicamente um vetor de 40 inteiros,
gravar neles seus próprios índices (0..39) e imprimir todos. Não esqueça de verificar falha de alocação
e liberar a memória ao final.

8) (Ponteiro não inicializado) Analise:

```
int *p;
*p = 3;
```

a) Qual é o problema?

b) Mostre duas formas de corrigir: (i) apontando p para uma variável existente; (ii) alocando memória.