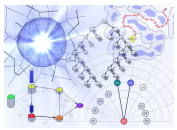


Recursividade-II

1. Codifique uma função recursiva que calcule a potencia de um dado numero, utilizando a seguinte formula:

$$x^n = \begin{cases} 1 & x = 0 \\ x & x = 1 \\ \left(x^{n/2}\right)^2 & x > 1,\ x \text{ is even} \\ x\left(x^{(n-1)/2}\right)^2 & x > 1,\ x \text{ is odd} \end{cases}$$

2. Escreva o output da seguinte função recursiva:

```
void ex(int n) {
        if (n <= 0)
                return;
        printf("%d",n);
        ex(n-2);
        ex(n-3);
        printf("%d",n);
}
```

3. Implemente uma função em liguagem C que calcule a distancia de Hamming entre dois vectores de inteiros, possivelmente de tamanhos diferentes:

   int Hamming(int *v1, int n1, int *v2, int n2);

4. Implemente uma função recursiva que imprima os primeiros "N" números de Fibanacci, endo estes elementos obtidos da seguinte forma:

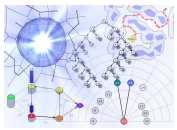
```
F(n)    = 1, se n = 1 ou n = 2
        = F((n+1)/2)² + F((n-1)/2)²     se n é impar
        = F(n/2 + 1)² - F(n/2 - 1)²     se n é par
```

5. Implemente uma função que imprima no écran todas as combinações possíveis de "n" elementos, "k" a "k":

   void printCombinations(int n, int k);

6. Considere o seguinte par de funções mutuamente recursivas. Imprima o output para a chamada g(g(2)) e calcule o respectivo valor de retorno:

```
int f(int n) {
        if (n == 0) return 0
        printf("f(%d)\n",n);
        return f(n-1) + g(n-1);
}
```



```
int g(int n) {;
        if (n == 0)
                return 0;
        printf("g(%d)\n",n);
        return g(n-1) + f(n);
}
```