

Questão 1 - Construa uma função recursiva que imprima números em ordem crescente, partindo de 0 até n. protótipo: void escreva_0_N(int n);

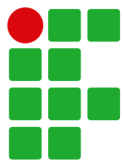
```
1. void escreva_0_N(int n){
2.     if(n == 0)
3.         return 0;
4.
5.     escreva_0_N(n - 1);
6.     printf("%i ", n);
7. }
```

Questão 2 - Construa uma função recursiva que imprima números em ordem decrescente, partindo de n até 0. protótipo: void escreva_N_0(int n);

```
1. void escreva_N_0(int n){
2.     if(n == 0)
3.         return 0;
4.
5.     printf("%i ", n);
6.     escreva_0_N(n - 1);
7. }
```

Questão 3 - Construa uma função recursiva que calcule e retorne a soma dos número de 0 até n. protótipo: int soma_N(int n);

```
1. int soma_N(int n){
2.     if(n == 0)
3.         return 0;
4.
5.     return n + soma_N(n-1);
6. }
```



Questão 4 - Construa uma função recursiva para calcular e retornar a potência de um número. protótipo: `int pot(int base, int expoente)`; Deste ponto em diante a definição do protótipo faz parte do exercício.

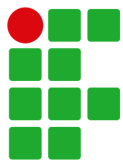
```
1. int pot(int base, int expoente){  
2.     if(expoente <= 1)  
3.         return 1;  
4.  
5.     return base * pot(n, expoente-1);  
6. }
```

Questão 5 - Construa uma função recursiva que calcule e retorne o fatorial de um número n.

```
1. int fatorial(int n){  
2.     if(n <= 1)  
3.         return 1;  
4.  
5.     return n * fatorial(n - 1);  
6. }
```

Questão 6 - Construa uma função recursiva que calcule e retorne o n-ésimo termo da sequência de Fibonacci.

```
1. int fibonacci(int n){  
2.     if(n<=2)  
3.         return 1;  
4.  
5.     return fibonacci(n-1) + fibonacci(n-2);  
6. }
```



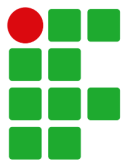
Questão 7 - Construa uma função recursiva que calcule e retorne a seguinte série:

$$\sum_{1}^n \left(\frac{1}{1} \right) + \left(\frac{1}{2} \right) + \left(\frac{1}{3} \right) + \dots + \left(\frac{1}{n} \right)$$

```
1. int serie_harmonica(int n){
2.     if(n <= 1)
3.         return 1;
4.
5.     return 1.0/n + serie_harmonica(n - 1);
6. }
```

Questão 8 - Construa uma função recursiva que retorne “verdadeiro” se um número n for primo e “falso” caso contrário.

```
1. int primo(int n){
2.     if(n <= 1)
3.         return 0;
4.
5.     return div(n, n-1);
6. }
7.
8. int div(int n, int i){
9.     if(i <= 2)
10.        return 1;
11.
12.    if(n % i == 0)
13.        return 0;
14.
15.    return div(n, i-1);
16. }
```

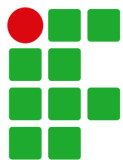


Questão 9 - Construa uma função recursiva que indique a quantidade de ocorrências de um dígito x em um número natural n . Exemplo: para $n = 522412$ e $x = 2$ a função deve retornar 3 ocorrências.

```
1. int seq(int val, int x){
2.     if(val == 0){
3.         if(x == val){
4.             return 1;
5.         }else{
6.             return 0;
7.         }
8.     }
9.
10.    int aux = val % 10;
11.    if(x == aux)
12.        return seq(val/10, x) + 1;
13.
14.    return 0;
15. }
```

Questão 10 - Construa uma função recursiva para calcular o máximo divisor comum de dois números (MDC). O MDC pode ser calculado a partir da seguinte definição recursiva: $MDC(x, y) = MDC(x - y, y)$, se $x > y$ $MDC(x, y) = MDC(y, x)$ $MDC(x, x) = x$.

```
1. int mdc(int a, int b){
2.     if(b == 0)
3.         return a;
4.
5.     return mdc(b, a % b);
6. }
```



Questão 11 - Construa uma função recursiva que receba um vetor de inteiros e o seu tamanho como parâmetro. A função deve *imprimir* os dados do vetor na tela.

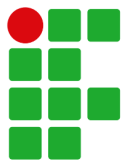
```
1. void print_vetor(int vet[], int tam){
2.     if(tam <= 1){
3.         printf(" %i |", vet[0]);
4.         return;
5.     }
6.
7.     print_vetor(vet, tam-1);
8.     printf("| %i ", vet[tam-1]);
9. }
```

Questão 12 - Construa uma função recursiva que faça a *leitura* dos elementos de um vetor de inteiros.

```
1. void scan_vetor(int* vet, int tam){
2.     if(tam == 0){
3.         scanf("%i",&vet);
4.         return;
5.     }
6.
7.     scanf("%i",&vet+tam);
8.     scan_vetor(vet, tam-1);
9. }
```

Questão 13 - Construa uma função que calcule e retorne a soma de todos os elementos de um vetor de inteiros.

```
1. int soma_vetor(int* vet, int tam){
2.     if(tam == 0)
3.         return *vet;
4.
5.     return *vet+tam + soma_vetor(vet, tam-1) ;
6. }
```

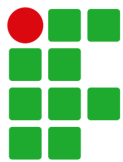


Questão 14 - Construa uma função recursiva que calcule e retorne a *soma* dos elementos pares presentes em um vetor de inteiros.

```
1. int soma_par(int* vet, int tam){
2.     if(tam == 0){
3.         if(vet[tam] % 2 == 0){
4.             return vet[tam];
5.         }else{
6.             return 0;
7.         }
8.     }
9.
10.    if(vet[tam] % 2 == 0){
11.        return vet[tam] + soma_par(vet, tam-1);
12.    }else{
13.        return 0 + soma_par(vet, tam-1);
14.    }
15. }
```

Questão 15 - Construa uma função recursiva que retorne o maior elemento presente em um vetor de inteiros.

```
1. int maior(int* vet, int tam){
2.     int aux;
3.
4.     if(tam == 0){
5.         aux = vet[0];
6.     }else{
7.         aux = maior(vet, tam-1);
8.         if(aux < vet[tam-1]){
9.             aux = vet[tam-1];
10.        }
11.    }
12.
13.    return aux;
14. }
```



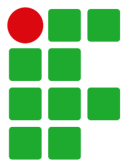
Questão 16 - Construa uma função recursiva que inverta os números de um vetor de inteiros.

```
1. void troca_index(int* vet, int id1, int id2){
2.
3.     int aux = vet[id1];
4.     vet[id1] = vet[id2];
5.     vet[id2] = aux;
6. }
7.
8.
9. int inverte_vetor(int* vet, int pos_inicial ,int tam){
10.
11.     if(pos_inicial < tam-1){
12.         troca_index(vet,pos_inicial,tam-1);
13.         return inverte_vetor(vet,pos_inicial+1,tam-1);
14.     }
15. }
```

Questão 17 - Construa uma função recursiva capaz de mostrar os dados de uma matriz de inteiros na tela.

```
1. void print_matriz(int mat[3][3], int lin, int col){
2.
3.     printf("%i\n", mat[col-1][lin-1]);
4.     if(lin == 1){
5.         if(col == 1){
6.             return;
7.         }
8.     }
9.     if(col == 1){
10.        col = 3;
11.        return print_matriz(mat, lin-1, col);
12.    }
13.    return print_matriz(mat,lin,col-1);
14. }
```

Questão 18 - Construa uma função recursiva que faça a leitura dos elementos de uma matriz.



```
1. void scan_matriz(int mat[3][3], int lin, int col){
2.
3.     scanf("%i\n", &mat[col-1][lin-1]);
4.     if(lin == 1)
5.         if(col == 1)
6.             return;
7.
8.     if(col == 1){
9.         col = 3;
10.        return scan_matriz(mat, lin-1, col);
11.    }
12.    return scan_matriz(mat, lin, col-1);
13. }
```

Questão 19 - Construa uma função recursiva que calcule e retorne a soma de todos os elementos da matriz.

```
1. int soma_matriz(int mat[3][3], int lin, int col){
2.
3.     if(lin == 1 && col == 1){
4.         return mat[0][0];
5.     }
6.
7.     if(col == 1){
8.         return (mat[lin-1][col-1] +
9.             soma_matriz(mat, lin-1, 3));
10.    }
11.    return (mat[lin-1][col-1] +
12.        soma_matriz(mat, lin, col-1));
13. }
```