01) Analise a complexidade dos seguintes algoritmos

a)

```c
int somaSimples(int x, int y){
    int s;
    s = x+y;
    if(s%2==0){
        return 1;
    }
    return 0;
}
```

b)

```c
int buscaBinaria(int arr[], int tamanho, int alvo){
    int esquerda = 0, direita = tamanho - 1;
    while (esquerda <= direita){
        int meio = (esquerda + direita)/2;
        if (arr[meio] == alvo){
            return meio;
        }
        else if (arr[meio] < alvo){
            esquerda = meio + 1;
        }
        else {
            direita = meio-1;
        }
    }
    return -1;
}
```

c)

```c
int soma(int a, int b){
    if(b==0){
        return 0;
    }
    if(b==1){
        return a;
    }
    return a + soma(a, b-1);
}
```

d)

```
void ordenarVetor(int *A, int tam_a){
    int pos_menor, aux;
    for(int i=0; i<tam_a-1; i++){
        pos_menor = i;
        for(int j=i+1; j<tam_a; j++){
            if(A[j]<A[pos_menor]){
                pos_menor = j;
            }
        }
        aux = A[i];
        A[i] = A[pos_menor];
        A[pos_menor] = aux;
    }
}
```

e)

```
void func(int n){
    for(int i=0; i<n; i++){
        printf("%d", i);
    }
}
```

02) Desenvolva cada código abaixo e determine a função analítica e complexidade big O de cada um

a) função recursiva que multiplique um dado inteiro A por B, usando somas sucessivas

b) função recursiva que calcule a divisão usando subtrações sucessivas

c) função recursiva que calcule o resto da divisão usando subtrações sucessivas

d) função recursiva que calcule a potência de um número usando multiplicações sucessivas

e) função recursiva que calcule a quantidade de caracteres de uma string

f) função recursiva que calcule o valor da serie S, descrita a seguir, para n>0:

s = 1 + 1/2! + 1/3! + ... + 1/n!