

**LISTA 4 - RECURSIVIDADE**

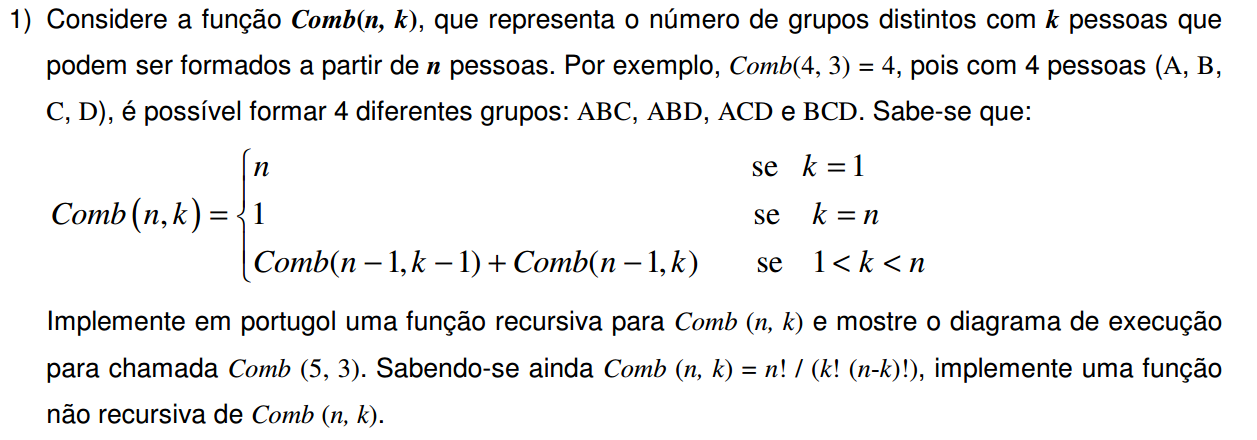
**Aluno**: Vitor Bruno de Oliveira Barth

**Professor**: Ruy de Oliveira

**Disciplina**: Algoritmos II

Cuiabá

2016



algoritmo **{ calcula uma combinação através de recursividade }**

**declare** n, k, comb

**leia** n, k

**imprima** *comb*(n, k)

**fim-algoritmo**

**subrotina** *comb*(n, k **numéricos**)

**declare** resultado numerico

**se** k = 1 **então** {caso base }

resultado <- n

**senão**

**se** k=n **então**

comb(1, 1) { se k = n, comb = 1 }

**senão**

resultado <- comb(n-1,k-1) + comb (n-1,k)

**fim-se**

**fim-se**

**retorna** resultado

**fim-subrotina**

algoritmo { resolve combinação através de fatorial }

**declare** n, k, comb

**leia** n, k

**imprima** *fatorial*(n)/(*fatorial*(k)\**fatorial*(n-k))

**fim-algoritmo**

**subrotina** *fatorial*(n **numéricos**)

se n = 0 então

resultado <- 1

**senão**

resultado <- n\*fatorial(n-1)

fim-se

**fim-subrotina**

**2 - Implemente recursivamente uma função Max que retorne o maior valor armazenado em um vetor V, contendo n números inteiros.**

algoritmo **{ retorne o maior valor de um vetor }**

declare i, n numerico

leia n

declare v1[n] numérico

leia v1

*max(*v1, n)

**fim-algoritmo**

subrotina *max*(v1, n numéricos)

declare resultado numéricos

se n = 2 então

resultado <- *maior*(v1[0], v1[1])

senão

resultado <- *maior*(v1[n], max(v1, n-1))

fim-se

retorna resultado

fim-subrotina

subrotina *maior*(a, b, numéricos)

declare resultado

se a > b

resultado <- a

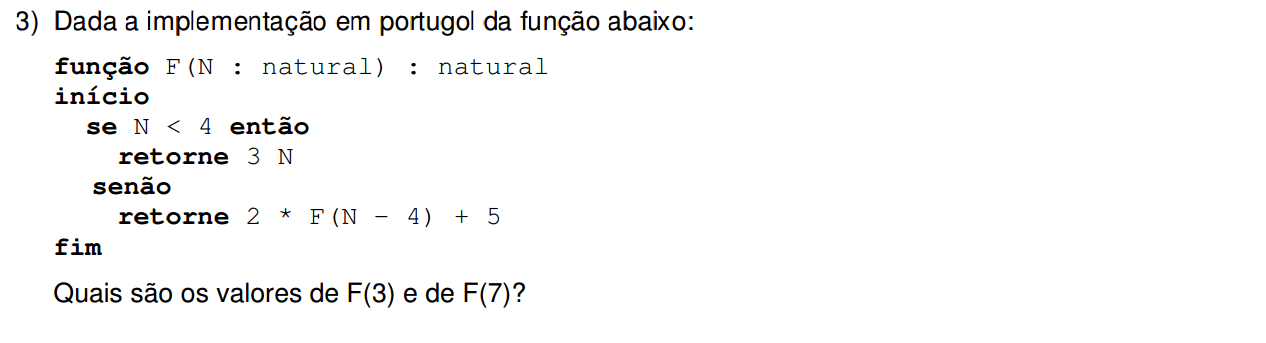
senão

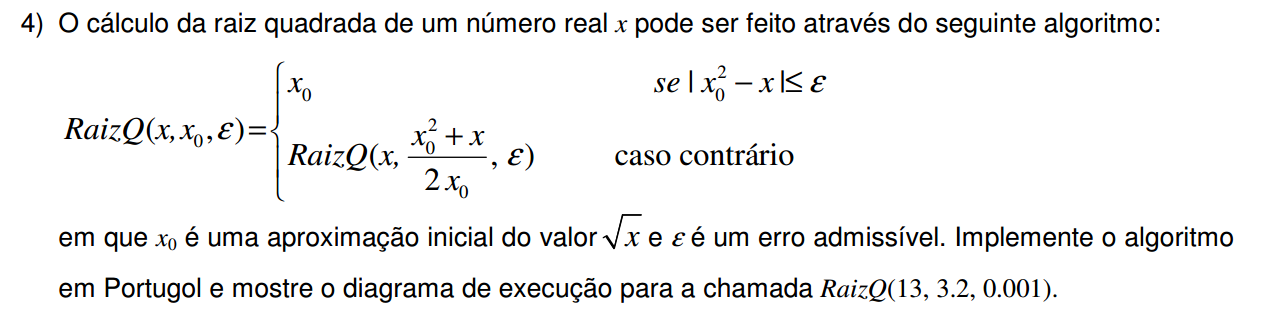
resultado <- b

fim-se

retorne resultado

fim-subrotina





algoritmo

declare raiz, aprox, erro numéricos

leia raiz, aprox, erro

imprima *raizq*(raiz, aprox, erro)

fim-algoritmo

subrotina raizq (r, a, e numérico)

se ((a\*a)-r) <= e então **{ caso base }**

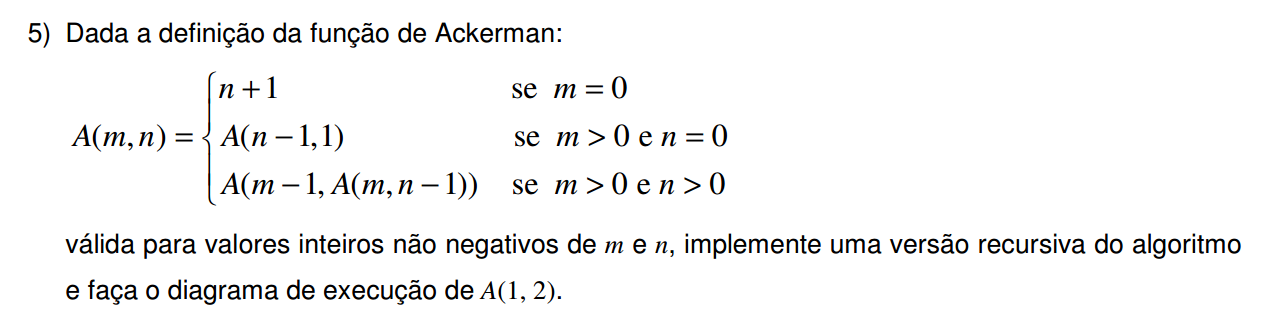
retorne a

senão

raizq(r, (((a\*a)+r)/(2\*a)), e)

fim-se

fim-subrotina



algoritmo

declare m, n, x numérico

leia m, n x

escreva *A*(m, n)

fim algoritmo

subrotina *A* (m, n numérico)

se m = 0 então **{ caso base }**

resultado <- n+1

senão se m > 0 && n = 0 então

resultado <- *A*(m-1, 1)

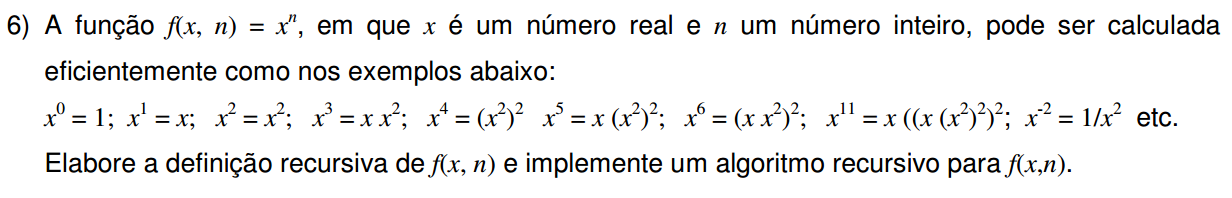
senão

resultado <- *A*(m-1, A(m, n-1))

fim-se

retorne resultado

fim-subrotina



algoritmo

declare x, n numérico

leia x, n

imprima f(x, n)

fim-algoritmo

subrotina f(x, n numérico)

declare resultado

se n < 0

resultado <- 1/*f*(x, n\*-1)

senão se n = 0 então

resultado <- 1

senão se n = 1 então

resultado <- x

senão se n = 2 então

resultado <- x²

senão se n%2 = 0 então

resultado <- *f*(fx, n/2), 2)

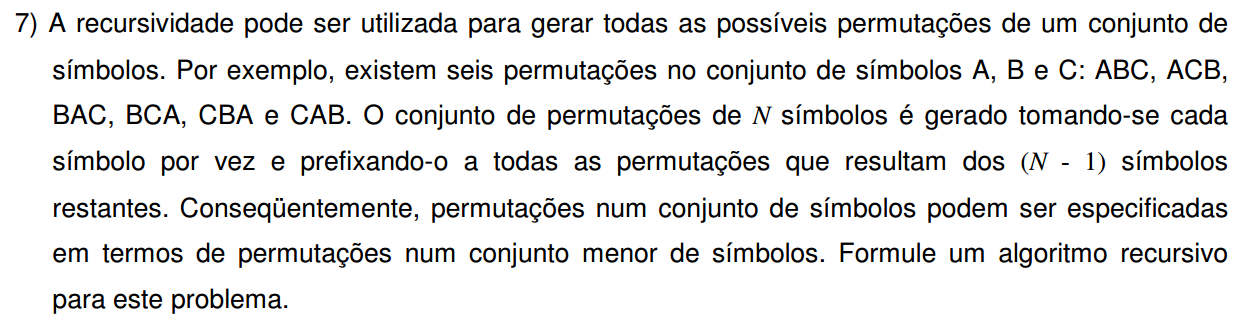
senão

resultado <- x\**f*(x, n-1)

**fim**-se

retorne resultado

fim-subrotina



algoritmo

declare num, i numérico

leia num

declare str[num] literal

leia str

*permut*(str, 0)

fim algoritmo

subrotina *permut*(str literal, k numérico)

declare i, tam numérico

tam <- *tamanho*(str)

se k = tam então

para i de 0 até num faça

imprima str[i]

fim para

senão

para i de k até tam -1 faça

*troca*(str, k, i)

*permut*(str, k+1)

*troca*(str, i, k)

fim para

fim se

fim-subrotina

subrotina *troca*(str literal, a, b numérico)

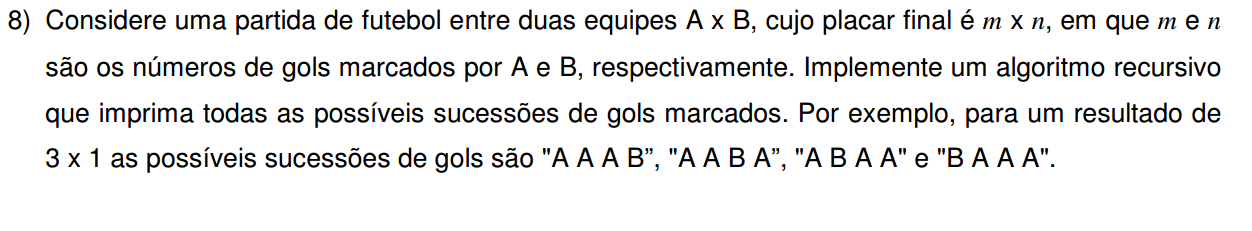
declare aux literal

aux <- str[a]

str[a] <- str[b]

str[b] = aux

fim-subrotina



algoritmo

declare golsT1, golsT2, i, j numérico

leia golsT1, golsT2

imprima calcula(golsT1, golsT2)

fim-algoritmo

subrotina *calcula*(golsT1, golsT2 numérico)

declare totalGols

totalGols <- golsT1+golsT2

se totalGols = 2

imprima AB, BA

senão

se golsT1 > 1

imprima A

para i de 0 a golsT1 faça

*calcula*(1, golsT2-1)

fim-para

senão se golsT2 > 1

imprima B

para i de 0 a golsT2 faça

*calcula*(golsT1-1, 1)

fim-para

fim-se

fim-se

fim-subrotina