**Função Arckermann**

**Python**

def ack(m, n):

return (n + 1) if m == 0 else (

ack(m - 1, 1) if n == 0 else ack(m - 1, ack(m, n - 1)))

print(ack(1, 4))

**Go**

package main

import "fmt"

func ack(m, n uint) uint {

if m == 0 {

return n + 1

} else if n == 0 {

return ack(m - 1, 1)

} else {

return ack(m - 1, ack(m, n - 1))

}

}

func main() {

fmt.Printf("%v", ack(1,4))

}

**Scala**

object Rextester extends App {

def ack(m: BigInt, n: BigInt): BigInt = {

if (m == 0) n + 1

else if (n == 0) ack(m - 1, 1)

else ack(m - 1, ack(m, n - 1))

}

println(ack(1,4))

}

**Node.js**

function ack(m, n) {

if (m == 0) {

return n + 1;

} else if(n == 0) {

return ack(m - 1, 1);

} else {

return ack(m - 1, ack(m, n - 1));

}

}

console.log(ack(1,4));

**Java**

class Rextester

{

public static long ack(long m, long n){

if(m == 0) {

return n + 1;

} else if(n == 0) {

return ack(m - 1, 1);

} else {

return ack(m-1, ack(m, n - 1));

}

}

public static void main(String args[])

{

System.out.println(ack(1, 4));

}

}

**JavaScript**

function ack(m, n) {

if (m == 0) {

return n + 1;

} else if(n == 0) {

return ack(m - 1, 1);

} else {

return ack(m - 1, ack(m, n - 1));

}

}

print(ack(1,4));