public class Counter {

private long c;

public Counter(){ c=0; }

public void incr(){ c+=1; }

public long count(){return c;}

public void add(long n){

if (n<0)return; // AGGIUNGE SOLO ELEMENTI POSITIVI

c+=n; }

public void reset(){c=0;}

public String toString(){ return ""+c; }} // END COUNTER

import java.util.Stack;

public class Iteration {

public static long st( int n, int k ) { // n, k > 0

long[] ct = new long[] { 0 }; // contatore: variabile di stato

sRec( n, k, 1, ct );

return ct[0]; }

private static void sRec( int n, int k, int q, long[] ct ) {

if ( (k == 1) || (k == n) ) ct[0] = ct[0] + q;

else { sRec( n-1, k-1, q, ct );

sRec( n-1, k, k\*q, ct );

} }

public static long stIter(int n, int k){

long[] ct = new long[]{0};

Stack<int[]> stack = new Stack<>();

int[] f = new int[]{n,k,1};

stack.push(f);

while (!stack.empty()){

f = stack.pop();

if ((f[1] == 1) || (f[1]==n)){ ct[0] = ct[0] + f[2];

} else { stack.push(new int[]{f[0]-1,f[1],f[1]\*f[2]});

stack.push(new int[]{f[0]-1,f[1]-1,f[2]});

return ct[0]; }

// \* @param n PRECONDIZIONE --> intero >= 0

// \* @return parte intera della radice quadrata di n

public static int intSqrt(int n){ // PREC: n>=0

int q=0, x=0, y=2\*q+1, z=n-y;

// INVARIANTI: 0<=q<=sqrt(n) ; x = q^2 ; y = 2q+1 ; y+z=n

while(x<=z){

q+=1;

x += (2\*q)-1;

y+=2;

z = z-2;

}return q; }}