We know, bin_coeff($l+1$, $l+1$) returns 1 by line 4.

By I.H., bin_coeff($l+1$, l) has maximum call stack depth l.

The function bin_coeff($l+1$, l) is also on stack, ∴ the max. call stack depth of bin_coeff($l+2$, $l+1$) is $l+1 = (k+1)$. □

∴ Maximum call stack depth of recursive_binomial_coefficient($k+1$, k) is $k = n-1$. ∴ Base case holds.

I.H: For $\forall l \in N$, $l < n$, the recursive_binomial_coefficient(l, k) function has a maximum call stack depth of $(l-1)$, for all $k < l$.

Induction Step: $l = n$. By line 7, we call: bin_coeff(n, k) = bin_coeff($n-1$, k) + bin_coeff($n-1$, $k-1$). By I.H., both bin_coeff($n-1$, k) and bin_coeff($n-1$, $k-1$) has a max. call stack depth of $(n-1)-1 = (n-2)$. ∴ bin_coeff(n, k) has a max. call stack depth of $(n-2)+1 = (n-1)$ □

∴ Space complexity of recursive_binomial_coefficient is: $c(n-1) = O(n)$

- It's easy to see that the iterative version takes up a constant space. The only unknowns are n, k both take 4 bytes each (constant).

$$\boxed{S_{\text{iterative\_binomial\_coefficient}}(n, k) = 0}$$

4) Determine the space complexity of the function created in Exercise 5, Sec-1.3 (pigeonhole principle)

Soln: The function int f(int x) has no structure variables, depending on input. x takes up constant space, the return address takes up const. space.

$$\boxed{S_f(I) = 0}$$