Write a program to find the number of ways to partition $a+b$ things into $a$ things in the left part and $b$ things in the right part. Include recursive time.

The number of ways to partition $a+b$ things into $a$ things and $b$ things is $(a+b)! / (a!b!)$. Here is a solution that is symmetric in $a$ and $b$.

```plaintext
if a=0 then x:= 1
else a:= a-1. x:= (a+b)! / (a!b!). a:= a+1. x:= x(a+b)/a fi
```

The assignment $x:= (a+b)! / (a!b!) / (a!b!)$ means $x' = (a+b)! / (a!b!)$ and $a' = a ∧ b' = b$. On the right side it is a recursive call. Stating it as an assignment makes the proof easy: just use the substitution law and simplify. The proof is by cases. First case:

$$a=0 ∧ (x:= 1) ⇒ (x:= (a+b)! / (a!b!)) \quad \text{definition of assignment}$$

$$a=0 ∧ x'= (a-1)! / ((a-1)!b!)(a+b)/a ∧ a'=a ∧ b'=b \quad \text{simplify}$$

$$a=0 ∧ x' = (a+b)! / (a!b!) ∧ a'=a ∧ b'=b \quad \text{specialization}$$

$$⇒ x:= (a+b)! / (a!b!)$$

Now the time.

$$t' = t+a \quad \text{if } a=0 \text{ then } x:= 1$$

```plaintext
else a:= a-1. t:= t+1. t' = t+a. a:= a+1. x:= x(a+b)/a fi
```

Proof by cases. First case:

$$a=0 ∧ (x:= 1) ⇒ t' = t+a \quad \text{definition of assignment}$$

$$a=0 ∧ x'= 1 ∧ a'=a ∧ b'=b ∧ t'=t ⇒ t' = t+a$$

Second case, starting with the right side:

$$a=0 ∧ (a:= a-1. x:= (a+b)! / (a!b!) ∧ a:= a+1. x:= x(a+b)/a) \quad \text{assignment}$$

$$a=0 ∧ (a:= a-1. x:= (a+b)! / (a!b!) ∧ a:= a+1. x'=x(a+b)/a ∧ a'=a ∧ b'=b) \quad \text{substitution law 3 times}$$

$$a=0 ∧ x' = (a+b)! / ((a-1)!b!)(a+b)/a ∧ a'=a ∧ b'=b \quad \text{simplify}$$

$$a=0 ∧ x' = (a+b)! / (a!b!) ∧ a'=a ∧ b'=b \quad \text{specialization}$$

$$⇒ x:= (a+b)! / (a!b!)$$

When refining $x:= (a+b)! / (a!b!)$, there was no time variable. Adding the time variable, we cannot write this as an assignment, because that would mean $t'=t$. We can put the result and the timing together as

$x' = (a+b)! / (a!b!) ∧ a'=a ∧ b'=b ∧ t'=t+a$

or as

$x:= (a+b)! / (a!b!). t:= t+a$

Here is a solution that is symmetric in $a$ and $b$.

```plaintext
if a=0 ∧ b=0 then x:= 1
else a:= a-1. b:= b-1. x:= (a+b)! / (a!b!). a:= a+1. b:= b+1. x:= x(a+b)(a+b-1)/(a+b) fi
```

And its execution time is smaller: $\min a b$. 

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Here is a solution with the same execution time and its recursion does not require a stack.

\[
x' = (a+b)! / (a!b!) \land t' = t + \min a b \iff \\
x := 1. \ x' = x \times a!b! / (a!b!) \land t' = t + \min a b
\]

\[
x' = x \times (a+b)! / (a!b!) \land t' = t + \min a b \iff \\
\begin{align*}
\text{if } &a=0 \lor b=0 \text{ then ok} \\
\text{else } &x := x/ab \times (a+b-1) \times (a+b). \ a := a-1. \ b := b-1. \ t := t+1.
\end{align*}
\]

\[
x' = x \times (a+b)! / (a!b!) \land t' = t + \min a b \fi
\]

Now, here is a for-loop solution. Define

\[
I_k = x = (a+k)! / (a!k!)
\]

Then

\[
x' = (a+b)! / (a!b!) \iff x := 1. \ I_0 \Rightarrow I'
\]

\[
I_0 \Rightarrow I' \iff \text{for } k := 0 .. b \text{ do } I_k \Rightarrow I'(k+1) \text{ od}
\]

\[
I_k \Rightarrow I'(k+1) \iff x := x \times (a+k+1) / (k+1)
\]

with timing \( t' = t+b \).

Finally, here are two functional solutions. Define

\[
f = \langle a, b : \text{nat} \rightarrow (a+b)! / (a!b!) \rangle
\]

Then

\[
f a b = \begin{cases} 
1 & \text{if } a=0 \\
f(a-1) b \times (a+b) / a & \text{else}
\end{cases}
\]

with execution time \( a \). For execution time \( \min a b \)

\[
f a b = \begin{cases} 
1 & \text{if } a=0 \lor b=0 \\
f(a-1)(b-1) \times (a+b-1) \times (a+b) / a / b & \text{else}
\end{cases}
\]