Problem 1. Translate, step-by-step, the following for-loop into a primitive recursive expression:

```c
int x = a + b;
for (int i = 1; i <= c; i++)
{x = x * b;}
```

You can use `add(., .)` (sum) and `mult(., .)` (product) in this expression. What is the value of this function when `a = b = c = 1`?

Problem 2. Translate, step-by-step, the following primitive recursive function into a for-loop:

\[ F = \sigma(PR(add(\pi_1^1, \pi_2^2), mult(\pi_4^4, \pi_5^5))). \]

For this function \( F \), what is the value \( F(2,0,1) \)?

Problem 3-4. Prove, from scratch, that the function \( f(a, b) = \frac{ab}{b^a} \) is primitive recursive. Start with the definitions of a primitive recursive function, and use only this definition in your proof – do not simply mention results that we proved in class, prove them.

Problem 5. Prove that the following function \( f(a, b) \) is \( \mu \)-recursive: \( f(a, b) = \frac{a^b}{b^a} \) when \( 0 < a < b \), and \( f(a, b) \) is undefined for all other pairs \((a, b)\). You can use the fact that vision and power are primitive recursive.

Problem 6. Translate the following \( \mu \)-recursive expression into a while-loop:

\[ f(a) = \mu m.(a^{m+1} > a). \]

For this function \( f \), what is the value of \( f(0) \)? \( f(1) \)?

Problem 7-8. Suppose that someone comes up with a new proof that not every computable function is primitive recursive, by providing three new examples of function \( N(n) \), \( N'(n) \), and \( N''(n) \) which are computable but not primitive recursive. What if, in addition to \( 0, \pi_k^k, \) and \( \sigma \), we also allow all three new function in our constructions? Let us call functions that can be obtained from \( 0, \pi_k^k, \sigma, N(n), N'(n), \) and \( N''(n) \) by using composition and primitive recursion \( 3\)-primitive recursive functions. Will then every computable function be \( 3\)-primitive recursive? Prove that your answer is correct.

Problem 9. Design a Turing machine for computing \( n + 3 \) in unary code. Trace it for \( n = 0 \).
**Problem 10.** Design a Turing machine for computing $n + 2$ in binary code. Trace it for $n = 0$. 