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

```java
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.

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

\[ F = \sigma(PR(mult(\pi_1^2, \pi_2^2), sum(\pi_4^4, \pi_2^4))). \]

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

Problem 3-4. Prove, from scratch, that the function \( f(a, n) = a^n / n! \) is primitive recursive, where \( n! \) stands for the factorial of \( n \), i.e., the product \( 1 \cdot 2 \cdot \ldots \cdot n \). 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 \( f(a, n) \) is \( \mu \)-recursive: \( f(a, n) = a^n / n! \) when \( n \leq 4 \), and \( f(a, n) \) is undefined for all other \( n \). You can use the fact that division and power are primitive recursive.

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

\[ f(a) = \mu n.(a^n / n! < 1). \]

For this function \( f \), what is the value of \( f(1) \)? \( f(2) \)? Take into account that \( 0! = 1 \) and \( a^0 = 1 \) for all \( a \).

Problem 7-8. Suppose that someone comes up with a new proof that not every computable function is primitive recursive, by providing a new example of a function \( N(n) \) which is computable but not primitive recursive. What if, in addition to \( 0, \pi_k \), and \( \sigma \), we also allow this new function in our constructions? Let us call functions that can be obtained from \( 0, \pi_k, \sigma \), and \( N(n) \) by using composition and primitive recursion \( N \)-primitive recursive functions. Will then every computable function be \( N \)-primitive recursive? Prove that your answer is correct.
Problem 9. Design a Turing machine for computing $n \div 3$ in unary code. Trace it for $n = 1$.

Problem 10. Design a Turing machine for computing $n \div 2$ in binary code. Trace it for $n = 1$. 