Test 1, Theory of Computations, Spring 2025

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

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

You can use $sum(.,.)$ and $mult(.,.)$ (product) in this expression.

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

$$F = \sigma(PR(mult(\pi^3_1, \pi^3_2), sum(\pi^5_5, \pi^5_2))).$$

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

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

Problem 6. Translate the following μ-recursive expression into a while-loop:

$$f(b) = \mu n.(n!/b^n > 1).$$

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

Problem 7-8. What if, in addition to 0, π^k_i, and σ, we also allow the function $A(A(n))$ in our constructions? Let us call functions that can be obtained from 0, π^k_i, σ, and $A(A(n))$ by using composition and primitive recursion *AA-primitive recursive* functions. Will then every computable function be AA-primitive recursive? Prove that your answer is correct.

Turn over, please

Problem 9. Design a Turing machine for computing negation $f(n) = \neg n$ in unary code: $f(0) = 1$ and $f(n) = 0$ for all $n > 0$. In other words:

- if the first symbol after the initial blank space is 1, we need to erase the number and go back;
- if the first symbol after the initial blank space is empty, then we need to place 1 there and go back.

Trace your Turing machine for $n = 1$.