Theory of Computations,
Test 1 for the course
CS 5315/CS 6315, Spring 2022

Name: ____

Up to 5 handwritten pages are allowed.

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

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

You can use add(.,.) (sum) and mult(., .) (product) in this expression.
What is the value of this function when a = b = c = 2?

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

$$f = \sigma(\sigma(PR(\sigma(0), add(\sigma(\pi^4{}_1), \pi^4{}_3)))).$$

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

3-4. Prove, from scratch, that the function f(p) = (p − 1)! / p, where a! is the factorial a! = 1 * 2 * ... * 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.

5. Prove that the following function f(p) is μ-recursive: f(p) = p! when p is either 1 or 2, and f(p) is undefined for other p.

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

$$f(a) = \mu m.(m * a > a).$$

For this function f, what is the value of f(0)? f(2)?

7-8. Suppose that someone comes up with a new proof that not every computable function is primitive recursive, by providing two new examples of functions N(n) and N'(n) which are computable but not primitive recursive. What if, in addition to 0, $\pi^k{}_i$, and σ, we also allow both new functions in our constructions? Let us call functions that can be obtained from 0, $\pi^k{}_i$, σ, N(n), and N'(n) by using composition and primitive recursion *2-primitive recursive* functions. Will then every computable function be 2-primitive recursive? Prove that your answer is correct.

9. Design a Turing machine for computing n + 4 in unary code. Trace it for n = 1.

10. Design a Turing machine for computing n + 4 in binary code. Trace it for n = 1.