Theory of Computations,
Test 1 for the course
CS 5315/CS 6315, Spring 2021

Name: _____

Up to 5 handwritten pages are allowed.

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

```
int x = p * q;
for (int i = 1; i <= r; i++)
   {x = x + q;}
```

You can use add(.,.) (sum) and mult(., .) (product) in this expression.
What is the value of this function when p = 1, q = 2, and r = 2?

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

$$f = \sigma(PR(mult(\pi^2_2, \sigma(0)), add(\sigma(\pi^4_1), \pi^4_3))).$$

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

3-4. Prove, from scratch, that the function f(p, q) = p % (q + 2) 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, q) is μ-recursive: f(p, q) = p / q when each of the values p and q is either 1 or 3, and f(p, q) is undefined for other pairs (p, q).

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

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

For this function f, what is the value of f(2, 4)? f(2, 5)?

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, π^k_i, and σ, we also allow this new function N(n) in our constructions? Let us call functions that can be obtained from 0, π^k_i, σ, 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.

9. Design a Turing machine for computing n + 2 in unary code. Trace it for n = 3.

10. Design a Turing machine for computing n + 2 in binary code. Trace it for n = 3.