Theory of Computations,
Test 2 for the course
CS 5315/CS 6315, Spring 2021

Name: ____

Up to 5 handwritten pages are allowed.

1. Why do we need to study decidable and recursively enumerable (r.e.) sets?

2. Is the intersection of two r.e. sets always r.e.? If yes, prove it, if no, provide a counterexample.

3. Is the set (A U B) − C, where X − Y is {x: x is in X and x is not in Y}, and A, B, and C are r.e., always r.e.? If yes, prove it, if no, provide a counterexample.

4. Prove that it is not possible, given a program that always halts, to check whether this program always computes 5n+6.

5. Design a Turing machine that computes n + 4 in binary code. Trace this machine on the example of $n = 101_2$.

6. Use a general algorithm for a Turing machine that represents composition to transform your design from Problem 5 into a Turing machine for computing f(f(n)) = n + 8.

7. Give a formal definition of feasibility. Give two examples:

- an example when an algorithm is feasible in the sense of the formal definition but not practically feasible, and
- an example when an algorithm is practically feasible, but not feasible according to the formal definition.

These examples must be different from the examples that we had in class.

8. What is P? NP? NP-hard? NP-complete? Brief definitions are OK. What do we gain and what do we lose when we prove that a problem is NP-complete? Explain one negative consequence (what we cannot do) and one positive one (what we can do).

9. What is propositional satisfiability? Give an example. Explain why this problem is important in software testing.

10. Step-by-step, apply the general algorithm to translate the following formula into DNF and CNF: $0.5 * a \geq b$.