Lecture #11: Multi-Tape Turing Machines, Nondeterministic Turing Machines, and the Church-Turing Thesis
Questions for Review

Multi-Tape Turing Machines

1. What is a ***multi-tape Turing machine***? Describe ways in which it is *similar* to the kinds of "Turing machines" that have already been defined, as well as ways that they are *different*.
2. Does the set of ***Turing-recognizable*** languages change if "standard" Turing machines are replaced by multi-tape Turing machines in the definition of this set of languages? Why (or why not)?
3. Does the set of ***Turing-decidable*** languages change if "standard" Turing machines are replaced by multi-tape Turing machines in the definition of *this* set of languages? Why (or why not)?
4. What is a ***simulation***? What does it generally include, and why is it useful?

Nondeterministic Turing Machines

5. What is a ***nondeterministic Turing machine?*** How is it different from the kind of (***deterministic***) Turing machine that has already been defined?
6. What does it mean for a nondeterministic Turing machine to ***accept*** an input string?
7. What does it mean for a nondeterministic Turing machine to ***reject*** an input string?
8. What does it mean for a nondeterministic Turing machine to ***loop*** on an input string?
9. Does the set of ***Turing-recognizable*** languages change if deterministic Turing machines are replaced by nondeterministic Turing machines in the definition of this set of languages? Why (or why not)?
10. Does the set of ***Turing-decidable*** languages change if deterministic Turing machines are replaced by nondeterministic Turing machines in the definition of this set of languages? Why (or why not)?

The Church-Turing Thesis

11. The “Church-Turing Thesis” is not a *theorem*. What other kind of thing is it, instead?
12. What does the Church-Turing Thesis assert? Why is this important, and how does it effect what we choose to study (and try to do) when we wish to learn about *computability*?