Ella, Yvonne, William

Worksheet #9
LBA, Post Tag, Counter Machines, Register Machines

1. What does the term **Turing-complete** mean?

A machine that can recognize all the same languages as a Turing Machine.

2. Consider the three models: **DFA**, **LBA**, and **TM**. For each model, write a sentence or two justification for why that model is the best one to use to reason about the capabilities of modern computers.

- A DFA has a finite number of states and memory like a computer, and that cannot be scaled arbitrarily, just like a computer.
- An LBA has a finite amount of memory, just like a computer. Also like in a computer, memory required for a string or problem scales based on the input. Although computers cannot scale their memory arbitrarily, because modern computers possess so much memory, most computers can adequately allocate memory to accommodate for demand.
- A Turing Machine has an infinite amount of storage and our computers have such a large amount of memory that it can be comparable. Also, if a computer were stuck in an infinite loop the program execution would never stop, just like when a TM never hits an accept or reject state. Finally, if memory limitations do not inhibit the computer, it can solve all the problems a turing machine can (it is turing-complete).

3. Can non-deterministic TMs accept languages that deterministic TMs cannot?

No. Given any finite string, a non-deterministic (ND) TM's processing of said string would have a finite number of branching results. Because we know this branching is finite and because Turing Machines have

infinite memory, we can always create a ND Turing Machine from a deterministic one, that stores all possible branchings on its tape and mimics the accept/reject behavior of the machine it is based on.

4. Why do you think a Queue automaton is Turing-complete, but a PDA is not?

A queue automaton is context-sensitive because you can dequeue a variable, use it, then enqueue it again, allowing you to continue to access the context as you move through the string, versus a stack where you either lose that variable or it is the only one you can look at (if you repeatedly pop and push it). This means that your context when using a PDA is only the most recent variable pushed onto the stack, which is arguably no context. In order for a machine to be Turing-complete, it must be able to process context-sensitive languages.

5. What is the difference between a context-free grammar and a context-sensitive grammar? Which one is more powerful?

Context free and context sensitive grammars have different production rules. In a context free grammar, the rules transform one variable to some subset of variables and terminal characters. However, in a context sensitive grammar, rules depend not only on the variable being transformed, but also on the variables and terminals adjacent to it. Context sensitive grammars are more powerful since more languages can be accepted and strings produced.

6. Are there any languages that a CFG can produce that a TM cannot accept? Why or why not?

We know CFGs and PDAs are computationally equivalent. We can easily represent the tape of the PDA and the stack of a PDA on 2 tapes in a TM. Therefore, anything that can be computed on a PDA could be computed on a TM if the instructions are converted accordingly, meaning for any language produced by a CFG, there exists a TM that accepts it.