1. Write a program called `dfa_equiv.cpp` that decides whether or not two DFA accept the same language. Your program will be given two filenames on the commandline. Each named file will contain the description of a DFA in a format similar to the one we used in the first assignment. Of course, since your program is only expected to work on deterministic automata, your input will never have more than one transition defined from the same state on the same input symbol.

Your program is expected to print out `[accept]` when the automata have the same language and print out `[reject]` when they don’t. Remember from class that we can answer the question of automata equivalence by using our construction for union, intersection and complement of automata languages. Here, however, the fact that inputs are deterministic simplifies this problem a little bit. Remember the cross product of states trick that we discovered in class when constructing the intersection automata for a pair of regular languages. I suggest using this trick for constructing both the intersection and union. Otherwise you will introduce nondeterminism and that will just make the problem more difficult.

2. Any finite language (a language containing only a finite number of strings) must be regular. Prove this.

3. Let $\Sigma = \{ a, b \}$. Consider the language of all strings that contain exactly three times as many occurrences of $b$ as occurrences of $a$. Call this language $L_{abbb}$. For example “abbb”, “babbabb” and “” (empty string) are in $L_{abbb}$, but “bab” “abba” and “bbababb” are not. Use the pumping lemma for regular languages to prove that $L_{abbb}$ is not regular.