Homework 2 (Due April 25 Tuesday)

Problem 1 (10pts)

In this question, we let *a* denote an arbitrary English letter and *d* denote a digit.

In a new language, identifiers can be formed by one of the following two ways: It can begin with ***a*** which is followed by zero or more characters, each being either ***a*** or ***d***. The second way to form an identifier is to begin with underscore "_", which is followed by *one or more* characters. Each character may be ***a***, ***d*** or underscore "_", but at least one of the character must either be an ***a*** or a ***d***.

(1.1) (15 pts) Please write a regular expression to represent the identifiers defined above.

(1.2) (10 pts) Please draw a DFA with minimal states to represent the identifiers defined above.

Problem 2 (15pts)

Write regular expressions for the following sets of characters:

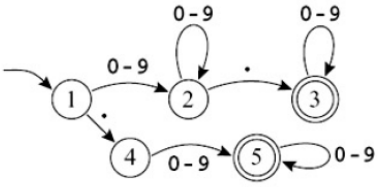
2.1 (5 pts) All strings of lowercase letters that contain the five vowels in order.

2.2 (5 pts) All hexadecimal numbers.

2.3 (5 pts) All strings of a' s and **b**'s with an even number of **a**'s and an odd number of **b**'s.

Problem 3 (15pts)

The following DFA is proposed as a specification for real numbers in a new programming language



REAL

3.1 Please list the sequence of states the input string “0334” will go through when it is fed to the DFA given above. Does the DFA accept the given input? Explain.

3.2 For the DFA given above, please write an equivalent regular expression.

Problem 4 (30pts) Please give the equivalent DFAs with minimal states to the following regular expressions.

4.1. (a*|b*)*   4.2. (a|b)*abb(a|b)*

Problem 5(30pts)

The grammar generates all even-length strings of a's.

S →aSa I aa

We can devise a recursive-descent parser with backtrack for this grammar. If we choose to expand by production S→aa first, then we shall only recognize the string aa. Thus, any reasonable recursive-descent parser will try S →aSa first.

Show that this recursive-descent parser recognizes inputs aa, aaaa, and aaaaaaaa, but not aaaaaa.