Name: _____ Date: _____

Note: The purpose of the following questions is:

• *Enhance learning*	• *Summarized points*	• *Analyze abstract ideas*

Class 07: Pumping Lemma Examples – Lex

Pumping Lemma:

The pumping lemma is difficult to understand and it is easy to go astray when applying it. Here are some common pitfalls. Watch out for them.

- One mistake is trying to use the pumping lemma to show that a language is regular. Even if you show no string in a language L can ever be pumped out, you can not conclude that L is regular.
- Another mistake is to start (usually inadvertently) with a string not in L.
- Finally, perhaps the most common mistake is to make assumptions about the decomposition `xyz`. The only thing we can say about the decomposition is what the pumping lemma tells us.

But even if you mater the technical difficulties of the pumping lemma, it may still be hard to see exactly how to use it. The pumping lemma is like a game with complicated rules. Knowing the rules is essential, but that alone is not enough to play a good game. You also need a good strategy to win.

1. Show that
$$L = \{vv^R : v \in \Sigma^*\} \qquad \Sigma = \{a, b\}$$
is not regular.
2. Show that
$$L = \{a^n b^l c^{n+l} : \ n, l \geq 0\}$$
is not regular.
3. Show that
$$L = \{a^{n!} : \ n \geq 0\}$$
is not regular.

An Application: Text Search - Finding Strings in Text

A common problem in the age of the Web and other on-line text repositories is the following. Given a set of words, find all documents that contain one (or all) of those words. A search engine is popular example of this process. The search engine uses a particular technology, called inverted indexes, where for each work appearing on the web (there are 100,000,000 different words), a list of all the places where that word occurs is stored. Machines with very large amounts of main memory keep the most common of these lists available, allowing many people to search for documents at once.

Inverted index techniques do not make use of finite automata, but they also take very large amounts of time for crawlers to copy the Web and set up the indexes. There are a number of related applications that are unsuited for inverted indexes, but are good applications for automation-based techniques.

4. *What are the characteristics that make an application suitable for searches that use automata?*

Nondeterministic Finite Automata for Text Search

5. Design an NFA to recognize occurrence of the words `web` and `ebay`. NFA is not a program, what are

your choices for implementing this NFA?

Lexical Analysis:
One of the oldest applications of regular expressions was in specifying the component of a compiler called a "lexical analyzer." This component scans the source program and recognizes all *token*, those substrings of consecutives characters that belong together logically. Keywords and identifiers are common examples of tokens, but there are many others.

The Complete Story for UNIX Regular Expressions

If you want get the complete list of operators and shorthand's available in the UNIX regular expression notation can find them in the manual pages for various commands. Thee are some differences among the various versions of UNIX, but a command like `man grep` will get you the notation used for the `grep` command, which in fundamental. "Grep" stand for "Global (search for) Regular Expression and Print," incidentally.

Exercises: Log in a UNIX machine and execute the following command: `$man grep`

Commands such as `lex` and `flex` have been found extremely useful because the regular-expression notation is exactly as powerful as we need to describe tokens. These commands are able to use regular-expression-to-DFA conversion process to generate an efficient function that breaks source programs into token.

Exercises:

6. **(Exercise):** Give a regular expression to describe phone numbers in all the various forms you can think of. Consider international numbers as well as the fact that different countries have different numbers of digits in area codes and in local phone numbers.
7. **(Exercise):** Give a regular expression to represent salaries as they might appear in employment advertising. Consider that salaries might be given on a per hour, week, month, or year basis. They may or may not appear with dollar sign, or other unit such as "K" following. There may be a word or words nearby that identify a salary. Suggestion: look at classified ads in a newspaper, or go on-line jobs listings to get idea of what patterns might b e useful.
8. **(Exercise)**: Explore an extended example of the sort of problem that arises in many Web applications. Suppose that we want to scan a very large number of Web pages to detect addresses. We might simply want to create mailing list. Or, perhaps we are trying to classify businesses by their location so that we can answer quires like "find me a restaurant with 10 minutes drive of where I am now"

Now let us focus on recognizing street addresses in particular. The entire expression we have developed for street addresses is

```
'[0-9]+[A-Z]? [A-Z][a-z]*( [A-Z][a-z]*)*
(Street|St\.|Avenue|Ave\.|Road|Rd\.)'
```

If we work with this expression, we shall do fairly well. However, there are some addresses we can not handle. What are we missing?

9. What are your comments about having a regular-expression compiler against a conventional programming language?