CS 631 Mid-term Exam
Spring 2015

Name: ____Ann Tupek_____

This exam is open book/notes/internet. Show all of your work for partial credit.

1. In designing a compiler, you will face many tradeoffs. What are the five qualities that you, as a user, consider most important in a compiler that you use? Provide a short justification (~ 1 sentence) for each one. How does that list change when you are a compiler writer? (15 points)

   As a user, the qualities that I want to see in a compiler are:
   - Data-flow analysis to do syntax highlighting and point out errors in the code before it is run, this way I can spot typo's and other syntax mistakes before running the program.
   - Coercion that will automatically convert an int to a float so that I can do something like 1+2.5 and get 3.5 without having to specifically type 1.0 or cast it myself.
   - The compiler should be correct in that it correctly translates the source language into the target language. If it isn't correct, then there's not much a point in using it.
   - The compiler should be fast. It should run as few passes as possible on the source code- I don't want to wait around forever while it runs.
   - The compiler should be portable, I want to be able to use the same compiler to compile code I wrote on my Mac on my Windows box, and vice-versa.

   This list changes in a few instances for a compiler writer. The optimizations for data-flow analysis may be difficult to implement and therefore might not be at the top of the list of items to complete. Also, making the compiler portable could be difficult in that the target machine code has to change depending on the operating system being used on the different machines. However, the list remains the same in that the compiler should be correct and it should be fast.

2. What is the difference between a lexeme and a token? Give an example where the parser needs the lexeme **and** the token. (10 points)

   A lexeme is a string of characters read by the lexer, and is used by the lexer to create its token. A token is what the lexer returns and sends on to the parser. The token contains its name and an attribute value. The attribute value is usually a pointer to the symbol table that provides more information about the token, such as its type and its value. The parser needs both the lexeme and the token when the lexeme is a string literal; its token is a literal and the attribute is the string literal.

3. Describe the language denoted by the following regular expression. (15 points)

```
a*ba*ba*ba*
```

The language is three b's with 0 or more a's at the beginning, end, and between each b.

4. The following grammar is not suitable for a recursive-descent parser. Identify the problem and correct it by rewriting the grammar. (15 points)

```
L -> R a
L -> Q b a
R -> a b a
R -> c a b a
*R -> R b c
Q -> b b c
Q -> b c
```

The line above marked with the * is left recursive. We need to change this and re-write it as right recursive. We can do that by changing the R productions to be:

R -> a b a D
R -> c a b a D
D -> b c D
D -> ∈

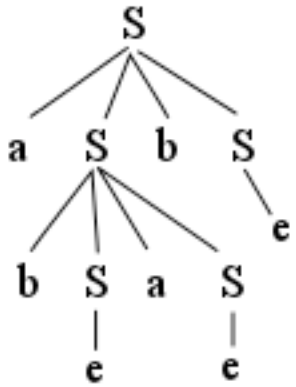
5. Compute First(L), First(R) and First(Q) for the grammar in question 4. (10 points)

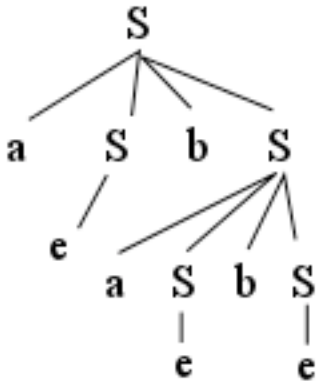
First(L) = First(R) & First (Q) = {a, c, b}
First(R) = {a, c}
First(Q) = {b}

6. Describe the language generated by the following grammar. Is the grammar ambiguous? Why or why not? (20 points)

```
S -> a S b S
S -> b S a S
S -> ∈
```

The language generated is any matched pairs of a's and b's. It is ambiguous because the string a b a b can be generated by multiple parse trees. The first parse tree expands the first S to a S b S and then expands the first S on the next level to b S a S, while the second parse tree also expands the first S to a S b S but then expands the second S on the next level to a S b S. Since we get two different parse trees for the same string, the grammar is ambiguous. (See parse trees below:)





7. Write a context-free grammar for the following regular expression. (15 points)

`((xy*x)|(yx*y))?`

S->x R x
  | y T y
  | ∈
R -> y R
  | ∈
T -> x T
  | ∈