Homework 4  10 points

Note, these exercises may be done in groups of one, two, or three. Working with someone else is strongly recommended. If more than one person is involved, list all the names on ONE set of answers. Groups may change throughout the term. Working in groups is a BIG plus for you. Take advantage of it. If you work in groups, you must work in the group for the ENTIRE assignment. It is considered cheating if you work with someone else for some of the answers, but turn in an individual copy of the answers. It is an all or nothing situation. You can’t work together on some questions and alone on some. Sometimes I see an individual whose name is listed in two groups. This is strictly forbidden and is considered cheating. You cannot work in two groups. Assignments are due at the beginning of class and should be typed when possible.

Chapter 4

1. For the NFA(non-deterministic finite automata) below, use the subset construction (discussed in class) to create a DFA(deterministic finite automata). Arcs labeled with e consume no input.

![DFA Diagram]

2. The following grammar needs to be massaged in order to be acceptable for a predictive parse table. In particular, left recursion needs to be eliminated and it needs to be left factored (see class notes if you need a refresher on these definitions). Show the modified grammar.

   S → T|U
   U → Ub | c
   T → aT | aX | z
   X → x

3. Consider the context sensitive example below:

   S → AB
   A → aAX | aX
   B → bBd | bYd
   Xb → bX
   XY → Yc
   Y → ε
Show the derivations of three different strings. Note, unless you are very lucky you will have the experience of getting to a place where you cannot get rid of the non-terminals. The only solution is to back up to your last decision and try something else. Hopefully you will see why context sensitive grammars are a pain. The “backtracking” to a previous decision is very expensive in terms of effort. Thus, modern day computer languages restrict themselves to constructs which can be recognized by a context free grammar.

4. Consider the grammar below. If it is ambiguous, demonstrate the ambiguity:

\[
\begin{align*}
A & \rightarrow BCD \\
B & \rightarrow bB \\
B & \rightarrow \emptyset \\
C & \rightarrow \text{Cg} \\
C & \rightarrow g \\
C & \rightarrow \text{Ch} \\
C & \rightarrow i \\
D & \rightarrow AB \\
D & \rightarrow \emptyset
\end{align*}
\]

5. Define three grammars G1, G2 and G3 as follows:
   - G1 has two rules: \( S \rightarrow aSb \) and \( S \rightarrow ab \) (a, b are terminals, S is the unique non-terminal and the start symbol.
   - G2 has five rules: \( S \rightarrow aSBC \), \( S \rightarrow abC \), \( CB \rightarrow BC \), \( bB \rightarrow bb \), \( bC \rightarrow bc \), \( cC \rightarrow cc \) (a, b, c are terminals, S, B, C are non-terminals and S is the start symbol.
   - G3 has three rules \( S \rightarrow aB \), \( B \rightarrow bS \), \( B \rightarrow b \) (a, b are terminals, S is the start symbol. S and B are non-terminals).

a. Classify each grammar as unrestricted, context-sensitive, context-free, or regular grammars.

b. Derive the string \( \text{aaabbb} \) using G1.

c. Derive the string \( \text{aaabbbccc} \) using G2.

d. Derive the string \( \text{abab} \) using G3.

---

**Chapter 5**

6. Consider this Pascal-like program in which each method has the following syntax: header, declarations (including declarations of local procedures), body:

```
program Example;
  var
    x: integer; y: float;
  procedure First;
    begin
      writeln(x, " ", y);
    end;
  procedure Second;
    var
      x: integer;
      y: string;
    procedure Third;
```
```haskell
var
 x:integer;
begin { Third }
 x := 99;
 First
end; { Third }
begin { Second }
 x := 88; y = "happy";
 Third
end { Second }
begin { Example }
 x := 77; y = 10.7;
 First();
 Second();
end. { Example }
a) What will be printed by this program if we assume static scope?
b) What will be printed by this program if we assume dynamic scope?

7. For the second type of symbol table (stack of symbol tables), show what the symbol tables look like at each of the points indicated in the C code below. In showing a symbol table, identify which variables and functions (and their types) are known. If there are multiple symbols tables active, show all of them. Assume brackets open/close a new scope.

```Haskell
int a,b     // global to entire program
int p(void)
{ int a,p;
  // local to scope of function p
  a=0 ;b=1;p=2 ;
  float b ;
  // local to unnamed scope
  a=3 ; b=4.0;
  print a+b
}
return p ;
}
main()
{float z;
 a  = p();
  //-------------- Point 3
 q();
}

Haskell
Complete sections 1, 2, 3 and 4 in the Learn You a Haskell tutorial.
http://learnyouahaskell.com/chapters

8. Give the list comprehension which produces a list of all multiple of 5 to 100.
9. What is the type of the tuple("Christopher", "Walken", 55).
10. What is produced by zip [1..] "abcde"
11. Write a function to compute the distance between two points.