1. Rewrite the following SDT:

\[
\begin{align*}
A & \rightarrow A \{a\} B \mid A B \{b\} \mid 0 \\
B & \rightarrow B \{c\} A \mid B A \{d\} \mid 1
\end{align*}
\]

so that the underlying grammar becomes non-left-recursive. Here, \(a\), \(b\), \(c\), and \(d\) are actions, and \(0\) and \(1\) are terminals.

2. Consider the following post system rules for type checking logical expressions over variables:

\[
\begin{align*}
\rho(\varepsilon) &= \tau \\
\rho \vdash e : \text{string} \\
\rho \vdash \#e : \text{int} \\
\rho \vdash e_1 / e_2 : \text{string}
\end{align*}
\]

Given \(\rho = \{\langle s, \text{string} \rangle \}\) prove that \(\rho \vdash \#("/\"s) : \text{int}\) with the method explained in class (and lecture notes).

3. Consider the type declaration

```c
struct BinTree
{
    int val;
    struct BinTree *left;
    struct BinTree *right;
};
```

Draw a graph that shows how the C compiler represents this type internally.

4. Consider the following C code:

```c
struct BinTree *p, q;
...
p->right->left = q.left;
```
Draw a type graph for \( p \) and \( q \) and indicate the types of \( p->right \), \( p->right->left \), and \( q.left \) that shows that the assignment is typed correctly (similar as shown on page 15 of slides for Ch6).

5. Consider the pseudo code program:

```plaintext
procedure P()  
  var a : real, 
      x : integer;  
procedure Q(a : integer)  
  var b : real;  
  procedure R(b : integer)  
  begin  
    Q(2*b);  
  end  
begin  
  ...  
  R(a-1);  
  ...  
end

begin  
  ...  
  x := 2;  
  Q(x);  
  ...  
end
```

(a) In the body of procedure \( R \), which variables and arguments are visible in scope (answer as in “\texttt{varname of procedurename}”)?

(b) Suppose \( P \) calls \( Q \), \( Q \) calls \( R \), and \( R \) calls \( Q \). Now, show the subroutine frames on the stack with the slots for locals and arguments and the access links (as in slide 14 of Ch.7 notes).

6. Consider Textbook 2nd ed. Figure 6.43 p.411 (Textbook 1st ed. pp.501–502). Assume that \( E_1 \) and \( E_2 \) are just the identifiers of variable names (as in Textbook 1st ed.). Give the parse tree with \texttt{truelist} and \texttt{falist} annotations for the example input \( a>b \ \&\& \ true \ || \ c<d \ || \ false \), assuming the emitted code starts at address 100. Give the backpatched code emitted for this example.