Programming Languages - Lab 3: Type Equivalence and Parsing

Name: YourName

1. [*20pts*] Here are some type and variable declarations in Pascal syntax.

type

 range = -5..5;

 table1 = array [range] of char;

 table2 = table1;

var

 x, y : array [-5..5] of char;

 z : table1;

 w : table2;

 i : range;

 j : -5..5;

State which **variables** are type equivalent under:

1. structural equivalence:
2. name equivalence:
3. declaration equivalence:
4. [*15pts*] Here is another example of type and variable declarations in Pascal syntax.

type

 rec1 = record

 x : integer;

 case boolean of

 true : (y : char);

 false : (z : boolean);

 end;

 rec2 = rec1;

 rec3 = record

 x : integer;

 case b : boolean of

 true : (y : char);

 false : (z : boolean);

 end;

var

 a, b : rec1;

 c : rec2;

 d : rec3;

State which **variables** are type equivalent under:

1. structural equivalence:
2. name equivalence:
3. declaration equivalence:
4. [*10pts*] Can a union in C, Modula-2, or Pascal be used to convert integers to reals and vice versa? Why or why not? (Hint: try this in C to see what happens).
5. [*15pts*] Left factor the following grammar.

S → Aa | B | D

A → Ct | Cu

B → Db | Dc | Cx

C → t | m

D → z | q | w

1. [*15pts*] Eliminate the left recursion from the following grammar.

S → Aa | Bb | c

A → Ab | d

B → ab | BcA

1. [*15pts*] Compute the FIRST and FOLLOW sets for the following grammar. The bolditems are terminal symbols.

P → D Q

Q → D Q | ε

D → **id** **=** E **;**

E → G F | **if** E **then** E **else** E | **func** **id** **.** E | **[**E E**]**

F → **op** E | ε

G → **id** | **con** | **(**E**)**

FOLLOW (P) =

FOLLOW (Q) =

FOLLOW (D) =

FOLLOW (E) =

FOLLOW (F) =

FOLLOW (G) =

FIRST (P) =

FIRST (Q) =

FIRST (D) =

FIRST (E) =

FIRST (F) =

FIRST (G) =

FIRST (P) =

FIRST (Q) =

FIRST (D) =

FIRST (E) =

1. [*10pts*] Is the grammar from the previous problem LL(1)? Explain.