Practice Problems: Top Down Parsing

Q1. Which of the following grammars are ambiguous?

a) {{S},{a,b},P,S}
   S → aSb|ab|ε
b) {{S},{a,b},P,S}
   S→SaSbS|ab
c) {{S},{+,-,*,(,),a},P,S}
   S→ S+|S-|S*|(S)|a|ε
d) {{S},{a},P,S}
   S→ S(S)|a| ε
e) {{S},{0,1},P,S}
   S→ 0S1S|1S0S|0|1
f) {{S,E,S'},{i,a,t,e,b},P,S}
   S→ iEtSS'|a
   S'→eS|ε
   E→ b

Q2. Write unambiguous grammar to the following grammar as per the following associativity and precedence order. The lowest precedence operators appear at the top, and the highest precedence operators appear at the bottom. Operators on the same line have the same associativity and precedence.

Left -associativity: * /
Right-associativity: + -
Left -associativity: ()

Grammar {{E},{+,-,*,/,(,),num},P,E}:

E→ E + E| E - E | E * E| E / E | (E) | num

Q3. Check whether the following grammar {{E},{+,-,*,/,(,),num},P,E} is suitable for the predictive parsing. If it is not, then convert the grammar into suitable grammar and construct the predictive parsing table. (Consider C language operator associativity and precedence order).

E→ E + E| E - E | E * E| E / E | (E) | num

Q4. Write a procedure for each of the non-terminals of this grammar{{D,F,S,E,A,T},{define, id, (,),{,}, return,int, float},P,D }using recursive descent parsing and parse the following input: define int id(int id,float id){ return id(id(),)}

D→define T id (F) {S}
F→T id , F | ε
S→ return E;
E→ id(A)

A→ ε| E , A
T → int|float

Q5. Write a procedure for each of the non-terminals of this grammar{{L,S},{if,bexpr,(,),else,while|{,},stmt,; },P,L}using recursive descent parsing and parse the following input: {if(bexpr){ while(bexpr) {stmt;}} else stmt;}

L→LS|S
S→if(bexpr) S|if(bexpr) S else S|while(bexpr) S|{L} | stmt;

Q6. Calculate the first and follow of all non-terminals in the following grammar

1. {{S,A,B,C},{a,b,g,h},P,S}.
   S→ACBS| CbB | Ba
   A→ daA|BC
   B→g|ε
   C→h|ε
2. {{R},{‘,*,(,),a,b,c},P,R}

   R→ R’ | ‘R |RR|R*|(R)|a|b|c

3. {{S,E,S’,E},{a,e,t,i,b},P,S}
   S→ iEtSS’|a
   S’→eS|ε
   E→ b
4. {{S,E}, {id, = , ; , if, (,) , while, ||,&&,rel, +,id true,flase }}
   S→id=E; | if (E) S |while (E) S| S S
   E→ E||E |E && E| E rel E |E + E| (E) |id| true |false

Q7. Eliminate the left recursion and write the first and follow for the following grammars

1. {{E,T,F},{+,-,*,/,(,),id},P,E}
   E→ E+T|E-T|T
   T→T*F|T/F|F
   F→(E) | id
2. {{S,B},{a,b,c,d},P,S}
   S→ Baa|bc|ab
   B →Bc|Sda|ε
3. {{S,A},{a,b,c},P,A}
   S→ AA |a|c
   A→ SSc| b
4. {{S,A},{a,b},P,S}
   S→ SAb| SA| b
   A→a

Q8. Left factor the following grammars

1. {{E},{i,:,=,e,(,),a,)}
   E→ i:=e | i(e)|a
2. {{E,T},{+,-,a,id,*}}
   E→ T + E| T – E | T |a
   T→ id * T | id
3. {{bexpr,bterm, bfactor},{or,and,not,true,false},P,bexpr}
   bexpr → bexpr or bterm | bterm
   bterm→ bterm and bfactor|bfactor
   bfactor →not bfactor |true |false
4. {{S},{a,b,c,d,e},P,S}
   S→ abd | ab| abcd |abcde|cd|e

Q9. Do the suitable changes to the below grammar {{S,T},{a , ^ , (,), , ,},P,S} and construct the predictive parsing table.

Obtain the moves for **(a,^)**

S→a|^|(T)

T→ T,S|S

Q10. Find whether the following grammar is LL(1) or not

S→ AB | PQx
A→ xy| m
B→bC
C→bC|e
P→pP|ε
Q→qQ|ε