Name - Syeda Rucha Qumrar
Roll no. - 14114802719
group - 5C7

Assignment - 2
(Compiler Design)

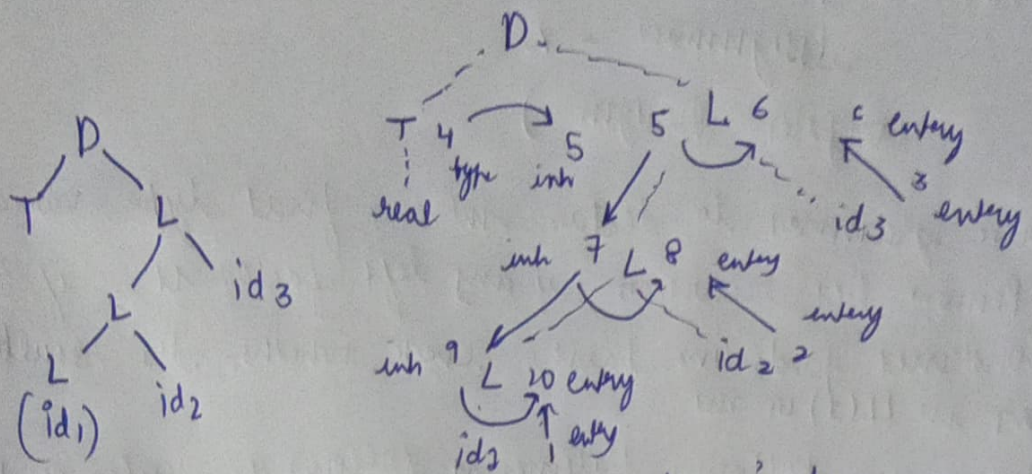
Ques 1. Write grammar to declar int or float type variables in C. Remove left recursion, if any left factor it, if required. Construct predictive parser & check whether the resultant grammar is LL(1) or no.

Check whether the following statements follows the rules or not
int a, b;  float a, int b;

Ans Non terminal D represents a declaration, which from production 1, consists of type T followed by a list of identifiers. T has one attribute, T.type which is the type in the declaration D. Non-terminal L also has one attribute, which we call it to emphasize that it is inherited attribute
The purpose of L rule is to pass the declared type down the list of identifiers, so that it can be appropriate symbol table entries, Production 2 and 3 each evaluate the synthesized attribute T.type, giving it the appropriate value, integer or float. This type is passed to attribute L.in in the rule for production 4.
Production 4 passes L.in down the parse tree. That is, the value L1.in is computed at a parse tree node by copying the value of L.in from parent of that node; the parent corresponds to the head of production. Production 4 and 5 also have a rule in which a function addType is called with 2 arguments
1. id.entry, a lexical value that points to a symtal table object.
2. L.in, the type being assigned to every identifier in the list.

The function addType properly installs the type L.in as the type of the represented identifier. Note that the side effect, adding the type info to the table, does not affect the evaluation order.

A dependency graph for the input string float id1, id2, id3 is shown below.




Given grammar is LL(1) as the intersection of firsts of the grammar are not ∅. Hence is parsed. Given strings are also parsed.

Ques 2. Test whether the grammar is LL(1) or not, and construct predictive parsing table for it.

$S \rightarrow AaAb \mid BbBa$
$A \rightarrow \varepsilon$
$B \rightarrow b \mid \varepsilon$

Step 1: No left recursion in the grammar, hence no modification required.

Step 2: Calculation of First Set

First(S) = First(AaAb) ∪ First(BbBa)
First(AaAb) = First(A) = ε
Since it contains ε, continue FIRST rule.
First(AaAb) = First(A) − ε ∪ First(aAb) = {a}
Similarly: First(BbBa) = {b}
First(S) = {a, b}

Step 3: Calculation of follow set:

Follow(S) = {$}
Follow(A) = First(aAb) = a
Follow(A) = First(b) = {b}
Follow(A) = {a, b}
Similarly Follow(B) = {a, b}

Step 4: Construction of Parsing Table: Grammar is LL(1)

| | a | b | $ |
|---|---|---|---|
| S | S → AaAb | S → BbBa | |
| A | A → ε | A → ε | |
| B | B → ε | B → ε | |

Nav Ques 3. Which operator has higher precedence?

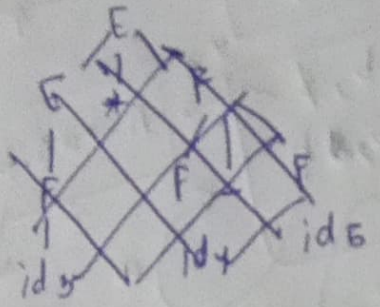
Rae $E \to E * F \mid F + E \mid F$

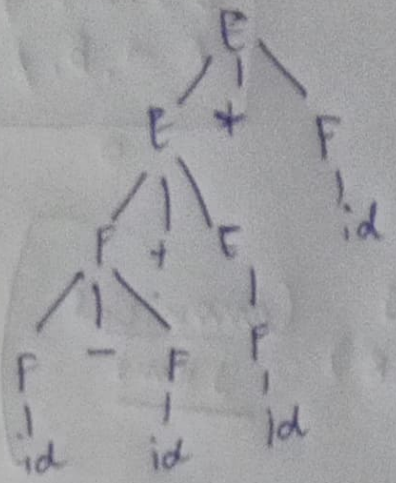
$F \to F - F \mid id$

Consider the expression "id - id + id * id"

As given $E \to E * F \mid F + E \mid F$

$F \to F - F \mid id$




Here we can see first + - will be evaluated then * is evaluated, so (-) has higher precedence than *.

The operator more close to leaf will be evaluated first as it has higher precedence, operator at root have the least precedence. If they are same level they have same precedence

| PRECEDENCE ORDER | | | | |
|---|---|---|---|---|
| − | > | + | > | * |
| (right) | | | | (left) |

Ques 4. Given the following CFG: $G_1 = \{\{S, A, B\}, \{a, b, x\}, P, S\}$

P: (1) $S \to A$ — $S' \to S$

(2) $S \to xb$

(3) $A \to aAb$

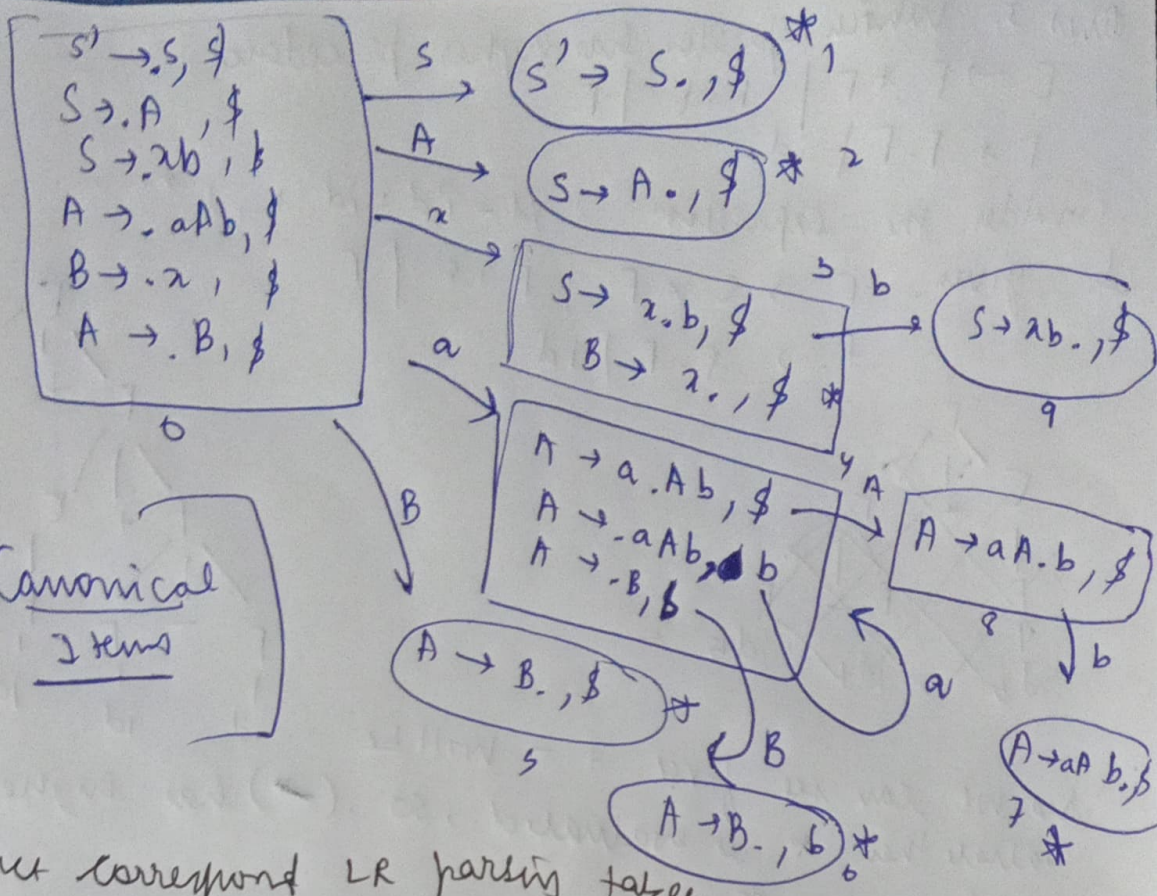
(4) $A \to B$

(5) $B \to x$

a) Compute the set of LR(1) items for this grammar and corresponding DFA.

| | First | Follow |
|---|---|---|
| S' • | {a, x} | {$} |
| S | {x, a} | {$} |
| A | {a, x} | {b} |
| B | {x} | {b} |

CLR / LR1




LR1 (CLR) Canonical Items

b) Construct corresponding LR parsing table

| | ACTION | | | | GOTO | | | |
|---|---|---|---|---|---|---|---|---|
| | a | b | a | $ | S' | S | A | B |
| 0 | S3 | | S4 | r2 r1 r5 r4 | | S1 | S2 | S5 |
| 1 | | | | accept | | | | |
| 2 | | | | r2 | | | | |
| 3 | | S9 | | r4 | | | | |
| 4 | | | S4 | | | | S8 | S6 |
| 5 | | | | r5 | | | | |
| 6 | | r5 | | | | | | |
| 7 | | | | r3 | | | | |
| 8 | | S7 | | r2 | | | | |
| 9 | | | | | | | | |

c) would this grammar be LR(0) why or why not

As is we can see from parse table . $ has multiple entries so its not a LR(0) grammer

d) We aab$ (Stack inputs and rules) 2)

e) would this grammar be suitable to be used parsed using top down LL parsing method? why?

Yes we can use this to parse our rules) grammars in top down method.