~~Assignment-6~~
Assignment :- 6.

1) Consider the following grammer and eliminate left recursion.

$A \rightarrow ABd / Aa / a$.

$B \rightarrow Be / b$.

Step :- 1 :- Remove direct left recursion from the non-terminal $A \rightarrow Aa\beta d / a\beta d / Aa / A$.

Step 2 :- factor out common prefix from the right hand side production.

$A \rightarrow aA$

$A \rightarrow aBdA' / BdA' / E$.

resulting grammer.

$A \rightarrow aA'$

$A' \rightarrow aBdA' / BdA / E$.

2) consider the following grammer and eliminate left recursion.

$A \rightarrow AAc / B$.

Step :- 1 $A \rightarrow AAc$

Step 2 :- non terminal left part.

$S \rightarrow AAc$.

Step 3 :- left recursion

A → S/A

step-4) factor out common terms from the right hand.

A → PD'

A → Sa/E

The resulting grammer will be after elimination of left recursion.

A → PD'

A' → Sa/E.

ii) S → bSSaaS/bSSaSb/bSb/a.

b S → aSSbS/aSaSb/abb/b.

To left factor this growth. left common for

1) S → bS'

2) S' → SSaaS/SaSb/Sb/E

3) S → a.

b) S → aSSbS/aSaSb/abb/b.

1. S → aS'

2. S' → SbS/SaSb/E.

3. S → abb.

4. S → b.

iv) check whether the following grammar is a.

Illustration.

S -> iEtS / iEtSeS / a.

E -> b.

find and follow sets.

1. first (S) = {i, a}

2. first (E) = {b}.

3. follow (S) = {$, e}

4. follow (E) = {t}.

check the condition for the given Gramar.

1. The first set of S = {i, a}.

2. The first set of E = {b} and follow set of E = {t}

Analytical: 1

① construct the recursive descent parser for the following grammer using.

$E \rightarrow TE'$

$E' \rightarrow TE'/E$

$T \rightarrow FT$

$T' \rightarrow +FT/E$

$F \rightarrow (E)/id.$

```
procedure E
{ T();
  E'();
}
```
$E \rightarrow TE'$

```
{ procedure E'()
{
if input symbol = '+' then.
  advance();
{ procedure T()
  f();
  T();
}
}
procedure T();
{ if input symbol = 'id' then
advance().
F();
T();
}
```
$T \rightarrow +FT.$

$T \rightarrow \phi FT.$

```
Procedure F()
{
if input symbol = id then
  advance()
check if input symbol = then;
  advance();
  E();
if input symbol = ')'
advance();
else error();
else error();
}
```

→ F → id

→ F → (E)

(2) Construct the following grammer for operation.
procedure parser.

solution.

	a	(	)	x	$
a		>	>	>	>
(	<	>	>	>	>
)	<	>	>	>	>
x	<	<	>	>	>
$	<	<	<	<	

operator procedure table.

step 1:-

$ (a(a,a)) $

we insert procedure operator the symbol.

$ \$ <(Ra>, <(a>, <a>)) \Rightarrow \$ $

step 2:-

we can scan and prune the string a's.

$ \$ < (<a>, <(<a>, <a>)) \Rightarrow \$ $

$ \$ <(S, <(<a>, <a>).>).>) \Rightarrow \$ $

$ \$ <(S, <(S, <a>)>).>) \Rightarrow \$ $

$ \$ <(S, <(S, S)>) \Rightarrow \$ $

$ \$ <(S, <(L, S) \Rightarrow) \Rightarrow \$ $

$ \$ <(S, <(L.)>) \Rightarrow \$ $

$ \$ <(L, S)> \Rightarrow \$ $

$ \$ <(L.)> \Rightarrow \$ $

$ \$ <(L)> \Rightarrow \$ $

$ \$ <> \Rightarrow \$ $

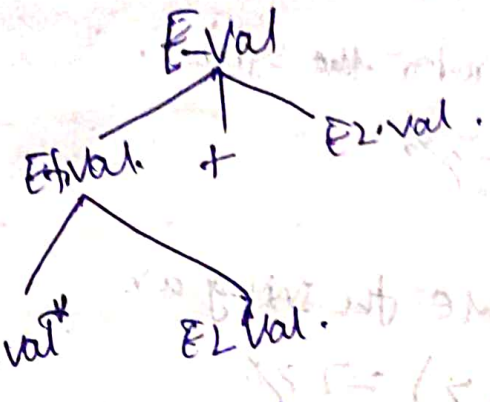
$ \$ \$ $

(2) Design the symbol graph for the following translation

$ E \rightarrow E1 + b. $

$ E \rightarrow E1 + E $

production	semantic problem.
$ E \rightarrow E1 + E2 $	Eval + E1.val + S.val.
$ E \rightarrow E1 b $	Eval → E1.val + E2.val.

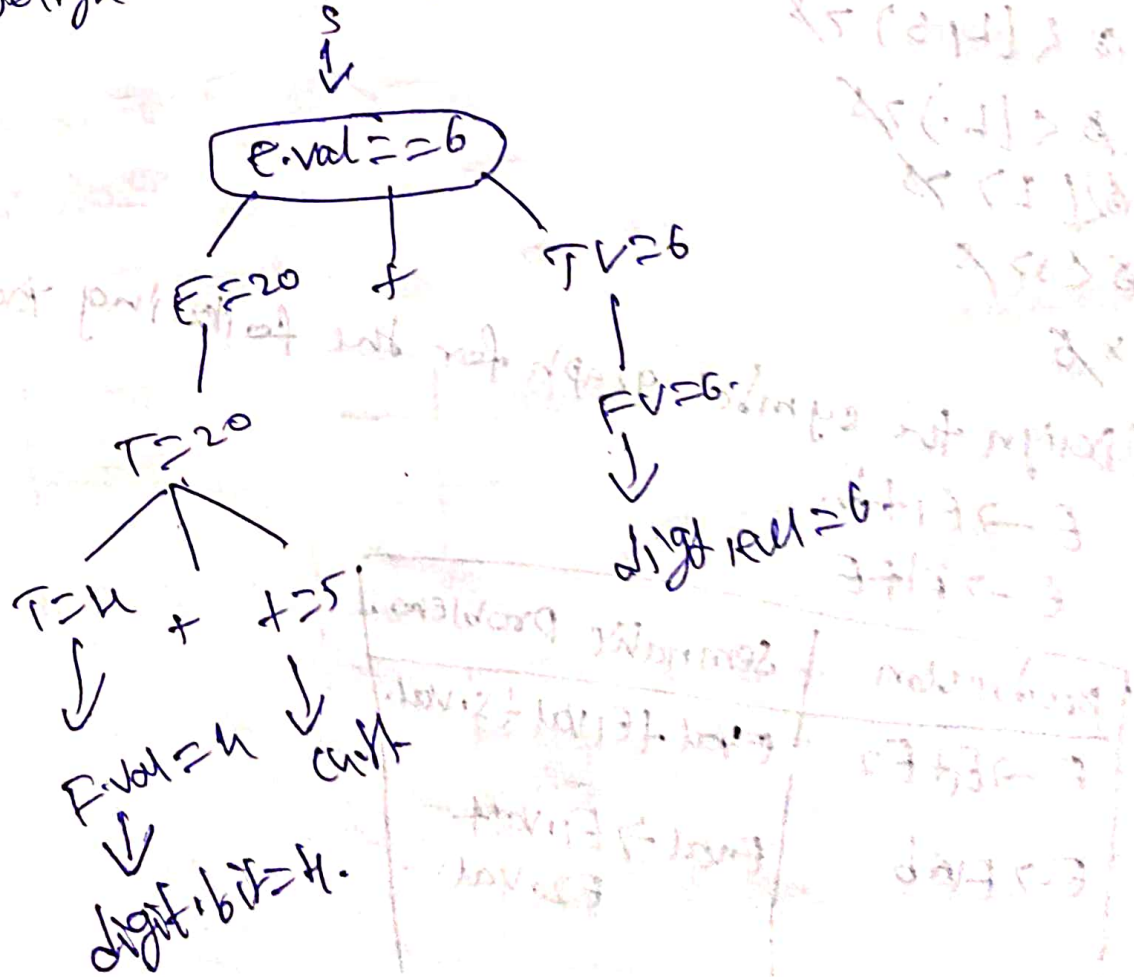


10) Design the dependency graph for the following grammer.

S -> T list
T -> int.
T -> char.
T -> double.

list -> list, id.
list -> id.

Design for above grammer of dependency graph



Analytical: 8.

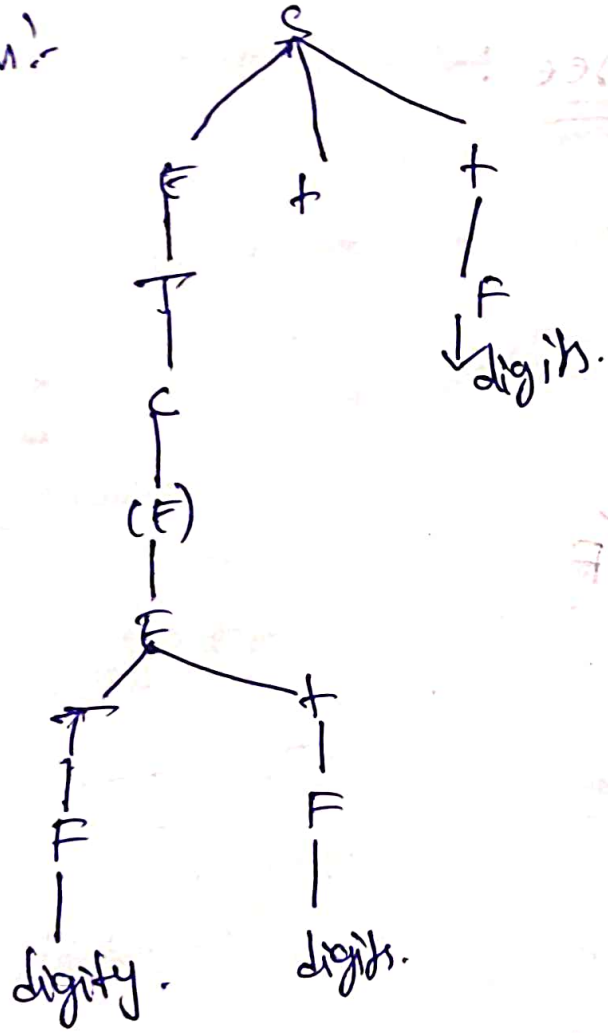
(1) Illustrate the SDD for simple digit calculation and show annoted parse tree for the expression (6+4)*(4+5)

$L \rightarrow E \& T \mid E - T \mid T$

$T \rightarrow T * F \mid T \mid F \mid F$

$F \rightarrow (E) \mid digit$

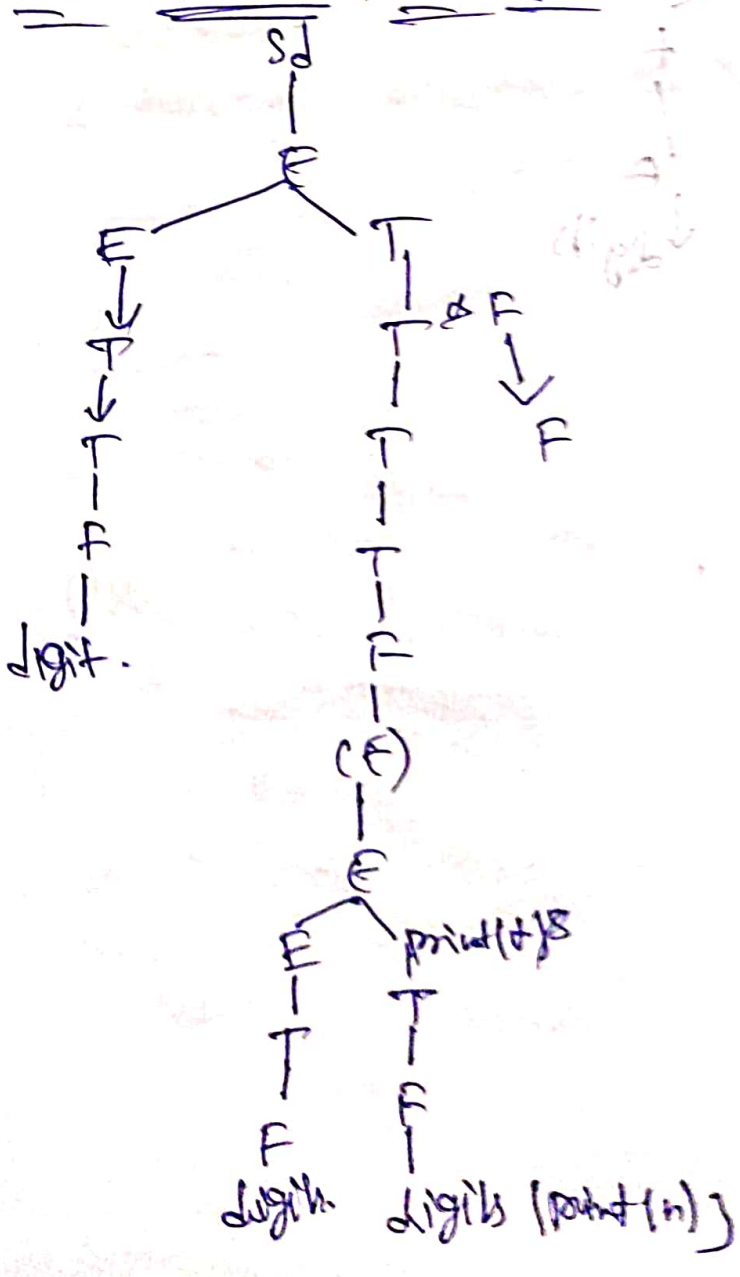
Ans:-



2. construct a decorated parse tree for the given input statement 4+5*5/2
SDD for sample put calculator.

1. S → E
2. E → T*F / T/F / F
3. T → T*F / T/F / F
4. F → (E) / digit

The decorated parse tree :-



(3) Give a support directed definition to differentiate expression formed by applying the arithmetic operator + and * to variable x. SDD: d(x+y)/dx.

Ans:- **SDD for Differentiation of Expression:**

S → E
E → E+T / T
T → T*F / F
F → x / constant.

S → E
|
E
|
T
| T*F
|
F
| x

∴ The derivation of the expression of $(3x^2 + x^4)$ with respect to x.

$x^4 (3^2x + x^2 x) + x$

(4) for the following given grammar construct the syntax directed definition and generate the code fragment.

S → EN.
E → E+T / E-T / T
T → T*F / T / F / F
F → (F) / digit.
N → ;

Ans:- SDD for the given grammar)

S → E print (unit)
E → E+T {E.val = E1.val + T.val} {E + {T
T → T*F {T.val = T1.val * F.val} T/F {T.val = F.val}
If. value {T.val · final}.

E -> { E.val + E.val } / to list.

N -> ;

Step 1:- Initialization the parsing stack

Stack [0]

Input = { 3 + 4 } ;

Step 2:- Applying LR applying action.

Step 3:- Apply the PLR parsing action for couple of this.

Step 4:- Reduce using rule & new Action

Stack [0]

Input = ;

Analytical : 9

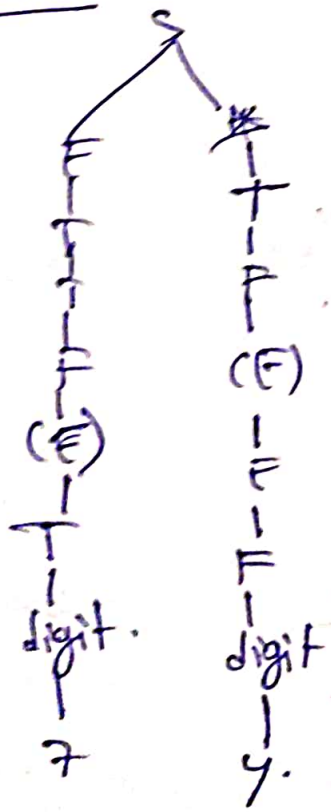
① Illustrate the SDD for simple desk calculation and showard expression (7+4)*(3+6).

S → En.
E → E+T/E.
T/K.
T → T*F/T/F/F.
F → (E)/digit.

SDD for simple desk calculation :-

1. S → E {print(E.val)}.
2. E → E+T {E.val = E1.val + T.val} H/T{E.val.Tal}.
3. T → T*F {T.val = T1.val * F.val} /T/F{Tval.
4. F(E){F.val = E.val}.

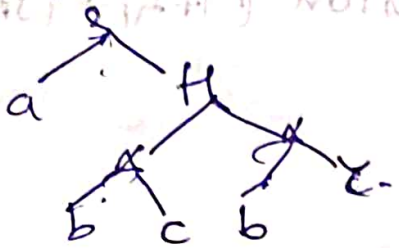
Expression : (7+4)*(3+6).



(2) Construct the syntax tree for the expression.

$a = b^* c + b^* c$.

The syntax tree for given expression is.



(3) Illustrate the bottom up evaluation for S attribute for the input $5*6+4$.

Given Grammer:-

$S \to EN$
$E \to E+T$
$E \to E-T$
$E \to T$
$T \to T*F$
$T \to T/F$
$T \to F$
$F \to E$
$E \to$ digit.

$S \to EN$.
$E \to E+T / E-T / T$
$T \to T*F / T/F / F$.
$F \to E$ | digit.

Bottom up evaluation!.

Input:- $5*6+4$.

1. Shift "5" onto the stack.
2. Shift "*" onto the stack.
3. Shift "6" onto the stack.
4. Shift "6" on the stack.
5. Reduce using $T \to$ digit.
6. Reduce using $T \to F$.
7. Shift "+" onto the stack.
8. Shift "4" on the shift.
9. Reduce using $T \to$ digit.
10. Reduce using $F \to T$.
11. Reduce using $S \to EE$.