31-07-23

Analytical day - 7

S:No = 33

R. praveen ganesh.

(1). Recursive descent parser

$E \rightarrow TE'$

$E' \rightarrow +TE' / \varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT' / \varepsilon$

$F \rightarrow (E) / id.$

(A)

```
procedure E()
Begin
  T()
  E'()
  End

procedure T()
Begin
  F()
  T'()
  End

procedure F()
Begin
if input_symbol = '+' then
  besign
      ADVANCE()
      T()
      E'()
  End

procedure T'()
  Begin
if input_Sy = '*' th
  begin
      ADVANCE()
      F()
      T'()
  End

procedu E()
  if input_symbol = 'id' then
    ADVANCE()
  Else if input_symbol = '('  then
    begin
```

E()

if input_symbol = 'g' then

ADVANCE()

Else ERROR()

End.

Else ERROR()

---

(2) Construct operation precedence parsing table

$S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

(A) $S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

1. Rows represent non-terminal (S & L)
2. Columns represent terminal (a, &,))

	a	,	)
S	r2		r2
L	r4	s3	r4

- sx : shift to state x
- rx : Reduce by production x.

(3) dependency graph

S:no = 33

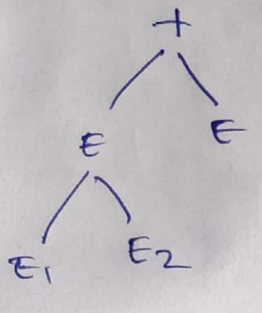
R. praveen ganesh.

$E \rightarrow E_1 + E_2$

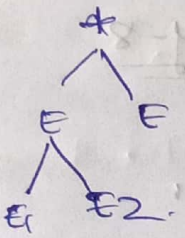
$E \rightarrow E | * E2$

(A)

$E \rightarrow E_1 + E_2$



$E \rightarrow E | * E2$



(4) Design dependency graph:

S → T List

T → int

T → float

T → char

T → double

List → List1, id

List → id

(A)

S → T List non terminal.

T → int : non-termin

graph

S.no = 33

R. praveen ganesh