Om Shete
TE C32
2103163

Experiment No : 9

Aim : Write a program to eliminate left recursion from the given grammar.

Theory :

- A production of the grammar is said to have left recursion if the leftmost variable of its RHS is same as variable of its LHS.
- A grammar $G(V, T, P, S)$ is left recursive because the left production in the form

  $A \rightarrow A\alpha \mid \beta$

- The above grammar is left recursive because the left of production is only at first position on right side of production.
- We can replace left recursion by replacing a pair of production with

  $A \rightarrow \beta A'$

  $A \rightarrow \alpha A' \mid \varepsilon$

- In left recursive grammar, expansion of A will generate $A\alpha$, $A\alpha\alpha$, $A\alpha\alpha\alpha$ on each side, causing it to enter into an infinite loop.
- Example :-

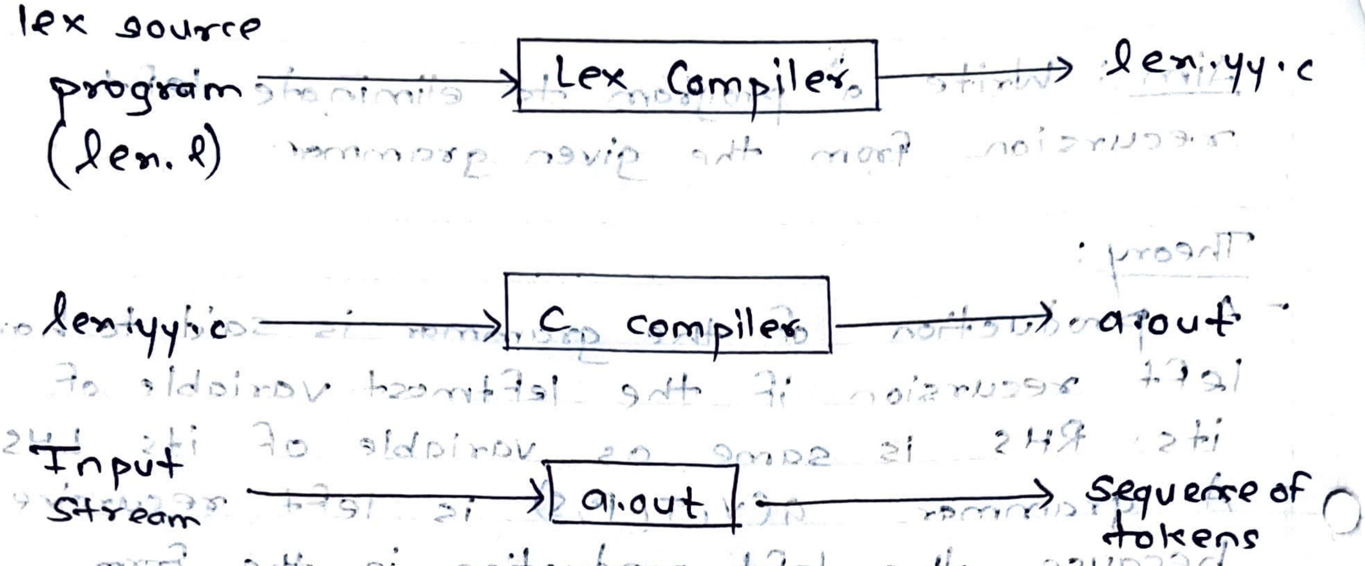
  $E \rightarrow E + T \mid T$

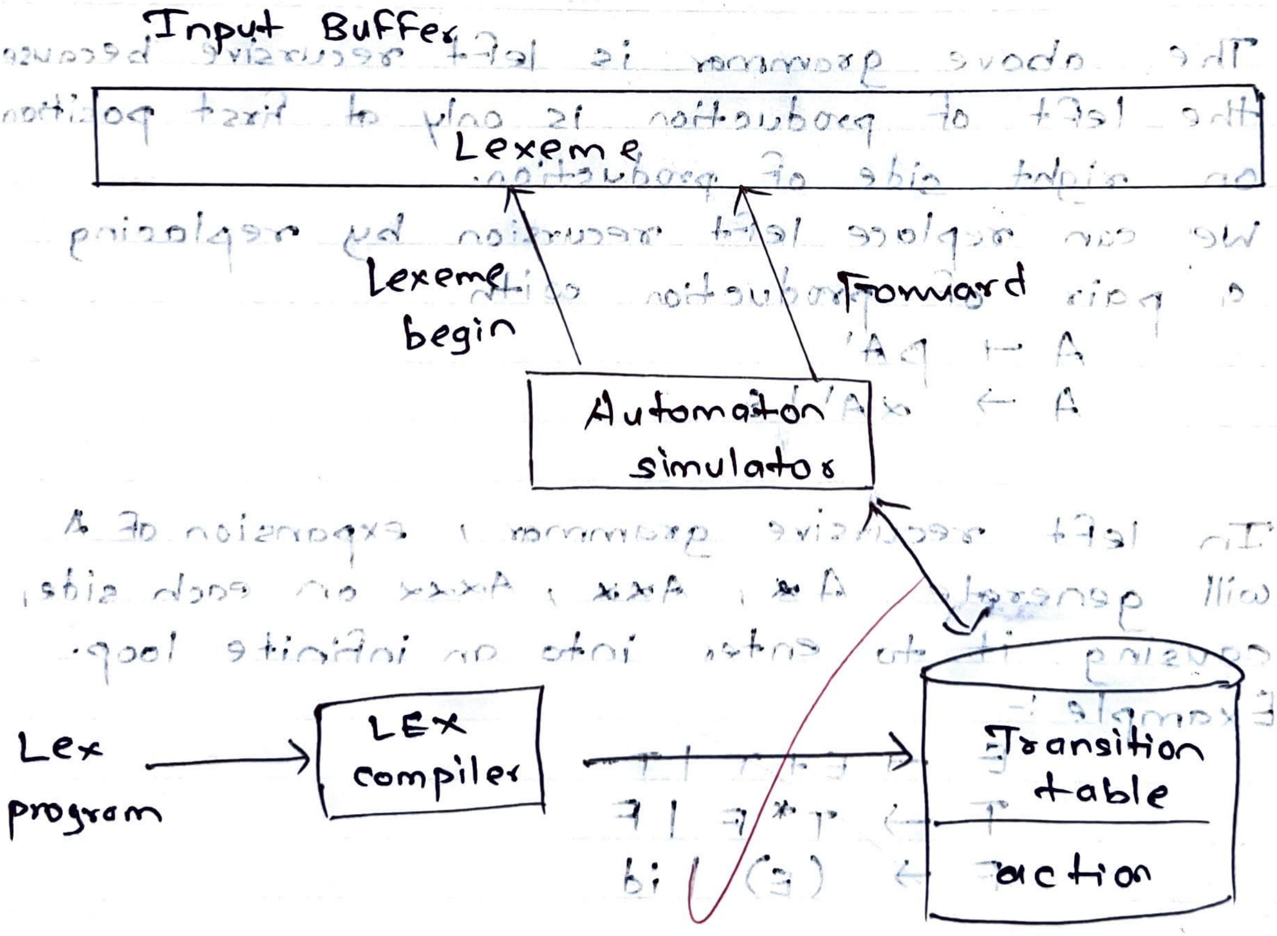
  $T \rightarrow T * F \mid F$

  $F \rightarrow (E) \mid id$

Compare $E \rightarrow E + T \mid T$

with $A \rightarrow A\alpha \mid \beta$


lex source program (lex.l)
Lex Compiler
lex.yy.c
lex.yy.c
C compiler
a.out
Input Stream
a.out
Sequence of tokens


Input Buffer
Lexeme
Lexeme begin
Forward
Automaton simulator
Lex program
LEX compiler
Transition table
action

$\therefore A = E$; $\alpha = +T$, $\beta = T$

$A \rightarrow A\alpha \mid \beta$ → change to → $A' \rightarrow \alpha A' \mid \varepsilon$

$\therefore A \rightarrow \beta A'$ means $E \rightarrow TE'$

Compare $T \rightarrow T{*}F \mid F$

$A = T$, $\alpha = +F$, $\beta = F$

$A = BA'$ means $T' \rightarrow {*}FT' \mid \varepsilon$

Production $F \rightarrow (E) \mid id$ does not have any left recursion.

$\therefore E \rightarrow TE'$

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

$T \rightarrow FT'$

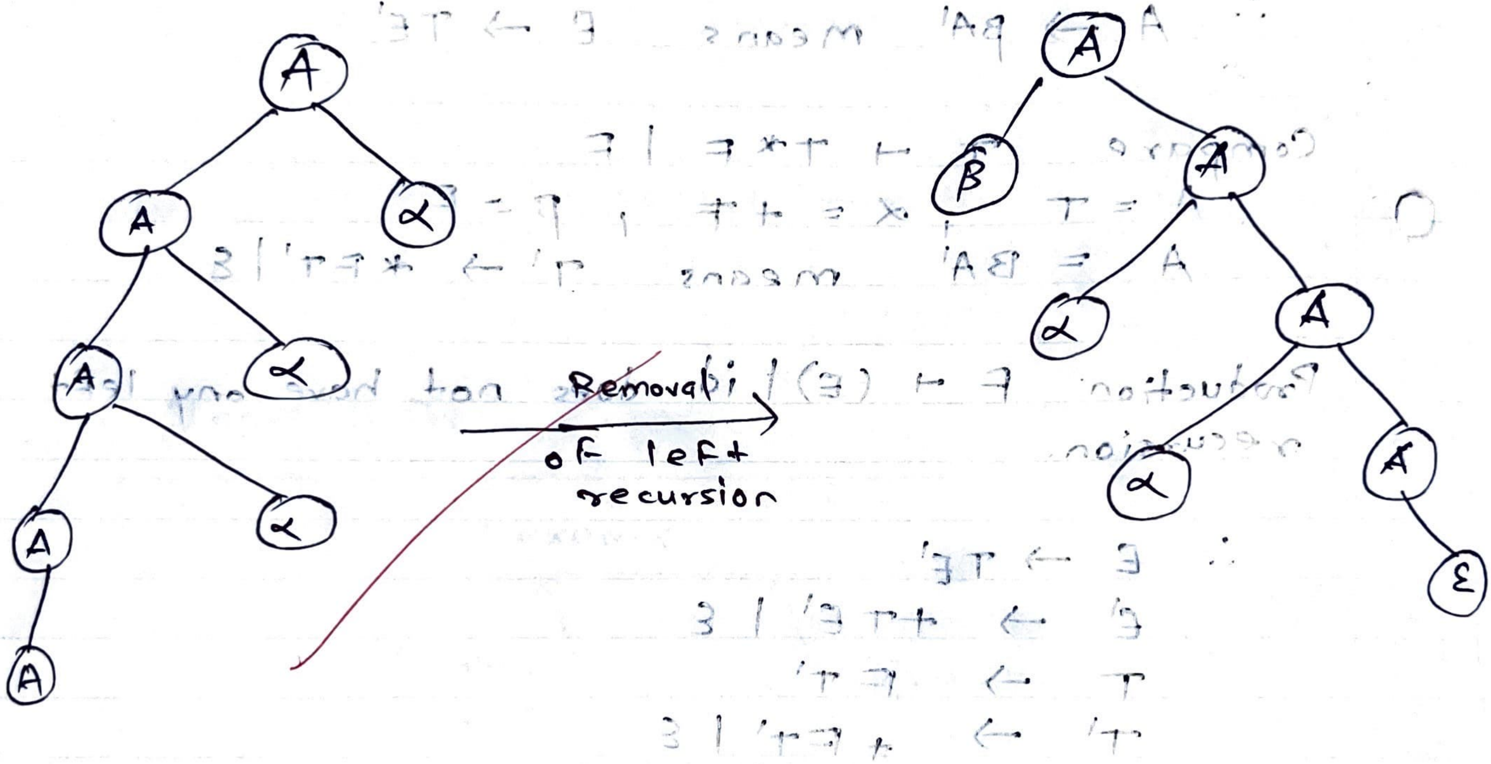
$T' \rightarrow +FT' \mid \varepsilon$

$F \rightarrow (E) \mid id$

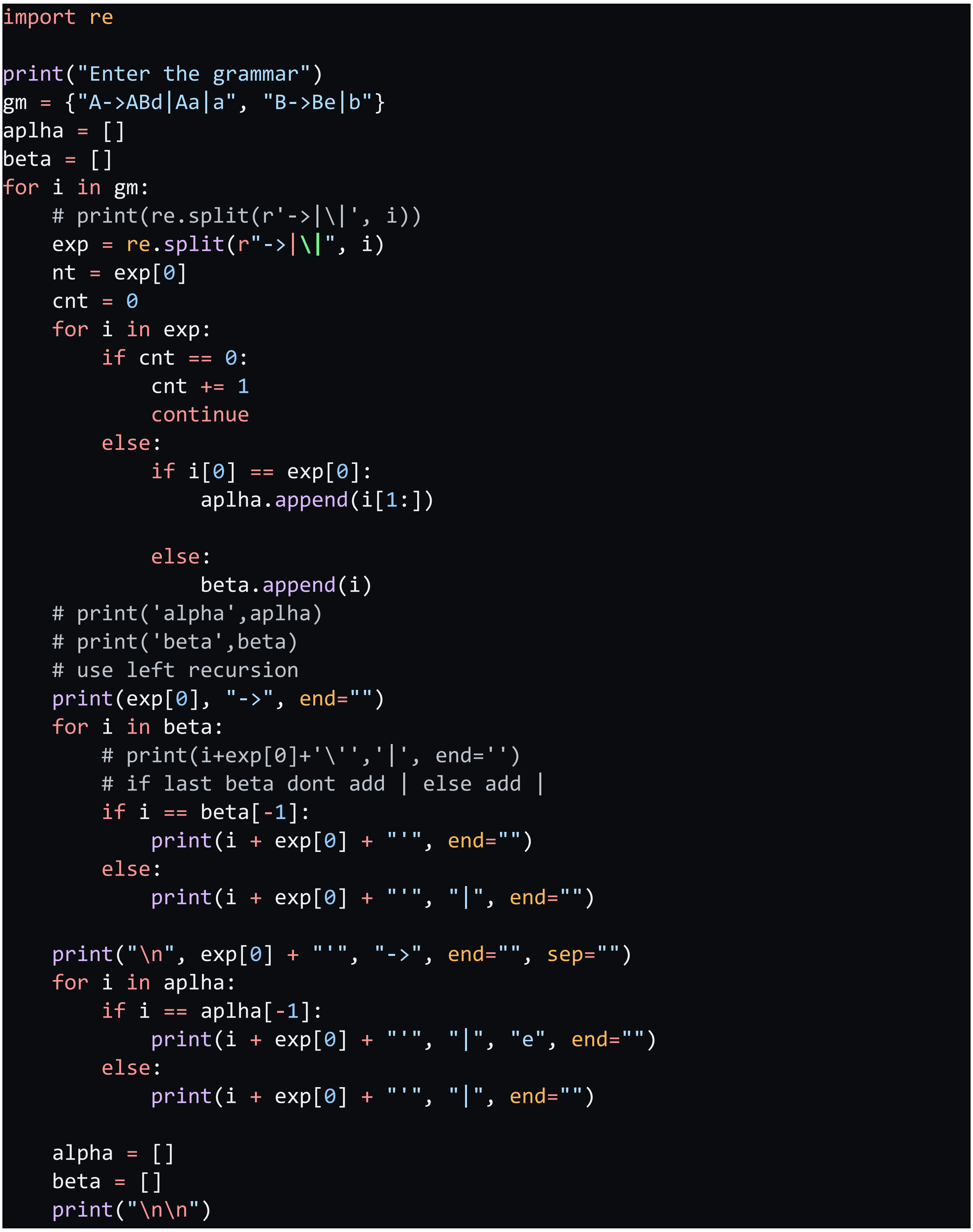
- Conclusion: In this experiment, we learn about left recursion and how to ~~manage~~ remove left recursion.

VA
28/03/2024

$$\boxed{A \rightarrow A\alpha \mid \beta} \xrightarrow[\text{left recursion}]{\text{Removal of}} \boxed{\begin{array}{l} A \rightarrow \beta A' \\ A' \rightarrow \alpha A' \mid \varepsilon \end{array}}$$



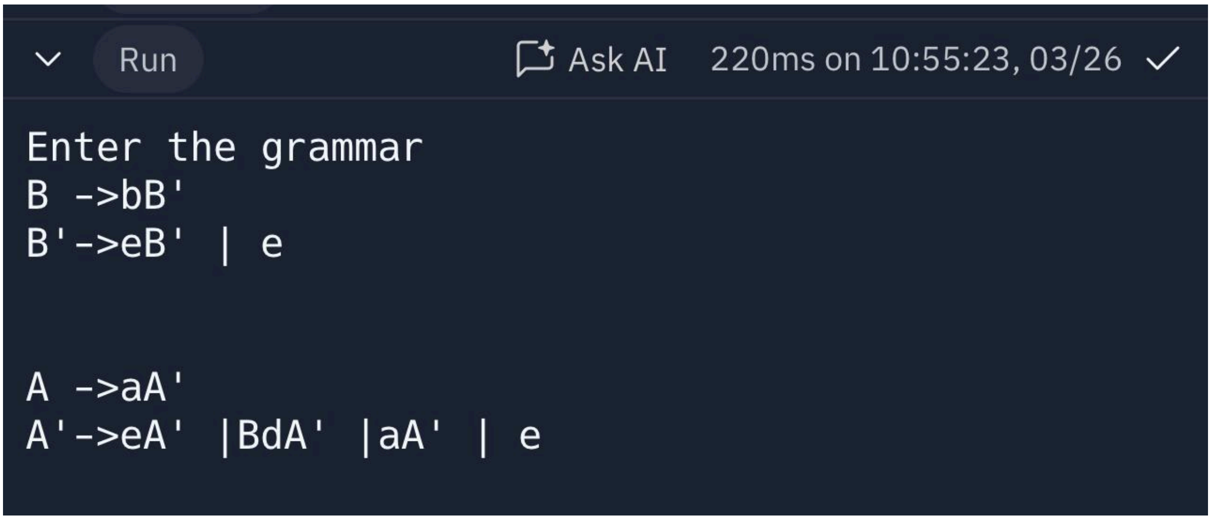
Code:



```python
import re

print("Enter the grammar")
gm = {"A->ABd|Aa|a", "B->Be|b"}
aplha = []
beta = []
for i in gm:
    # print(re.split(r'->|\|', i))
    exp = re.split(r"->|\|", i)
    nt = exp[0]
    cnt = 0
    for i in exp:
        if cnt == 0:
            cnt += 1
            continue
        else:
            if i[0] == exp[0]:
                aplha.append(i[1:])

            else:
                beta.append(i)
    # print('alpha',aplha)
    # print('beta',beta)
    # use left recursion
    print(exp[0], "->", end="")
    for i in beta:
        # print(i+exp[0]+'\'','|', end='')
        # if last beta dont add | else add |
        if i == beta[-1]:
            print(i + exp[0] + "'", end="")
        else:
            print(i + exp[0] + "'", "|", end="")

    print("\n", exp[0] + "'", "->", end="", sep="")
    for i in aplha:
        if i == aplha[-1]:
            print(i + exp[0] + "'", "|", "e", end="")
        else:
            print(i + exp[0] + "'", "|", end="")

    alpha = []
    beta = []
    print("\n\n")
```

Output:

Om Shete - C3 - 2103163



```
Enter the grammar
B ->bB'
B'->eB' | e


A ->aA'
A'->eA' |BdA' |aA' | e
```