1705045

Iftekhar Hakim Kaowsar

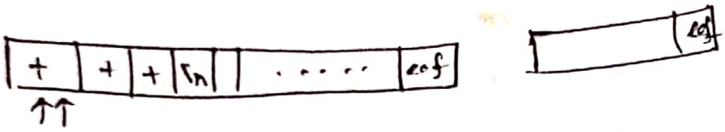
(2) If the code optimizer changed the result produced by the program, it is not acceptable at all. An optimizer will work just to make things faster or even simpler. But, it can never change the result. It will be treated as wrong code optimizer.

(1) After lexical analysis, code may need to be preserved. Because, after this phase, to give better error signals, we somptimes check characters (that may be changed to overcome error). Although this things, are expensive, such operation may need main code again.

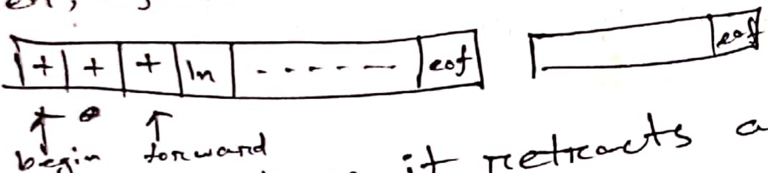
(3) I support the 2nd opinion and discard others. Because, for (1), it is not impossible. Rather, it is dependant on implementation of designer. A designer ~~may~~ possible can design a compiler where a seperate thing for lexical analysis does not exists. He may use an altogether complex designing. And, (iii) is not correct, of course, some measure should be taken.

P-2
1705045

(4)

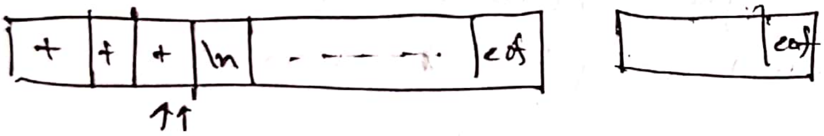


Then, after 2 step

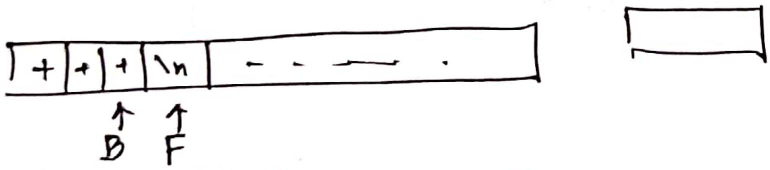


After reaching here, it retracts and accept "++", returns to parser. <++>

Again,

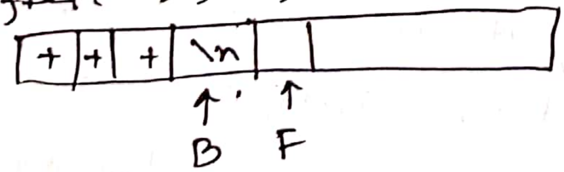


Then



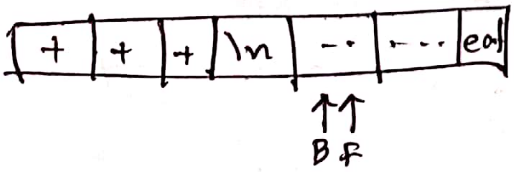
Returns "+". or <+>

After 2step,



Returns \n

Then,



This continues until eof for code end is found.

P- 3
1705045

⑤

S → (L) | a
L → sL , s | ϵ

~~FIRST {L} = {S}~~
FIRST {S} = {(, a}

```
void S () {
switch (lookahead) {
    case '(' :
        match('(');
        ~~match (L);~~ L();
        match (')');
        break;
    case a :
        match(a);
        break;
   default :
       go("error syntaxically");
 }
}
void L () {
   switch (lookahead) { ~~go_empty(); break;~~;
      ~~case empty:~~
      case S :
          match(S);
          L()
          match(,)
          S(); break;
    \\no error for default
   }
}
```

```
void match(t) {
if (lookahead==t)
   lookahead.next();

else
   go("error");

}
~~void go_empty()~~ {
   ~~return~~;
```

P-9
1705045

⑥ $L \rightarrow L, cd \mid ab$

will be chonge to,

$L \rightarrow abL'$

$L' \rightarrow ,cdL' \mid \epsilon$