LR(0)

Aim: A program to implement LR(0) items

Algorithm:-

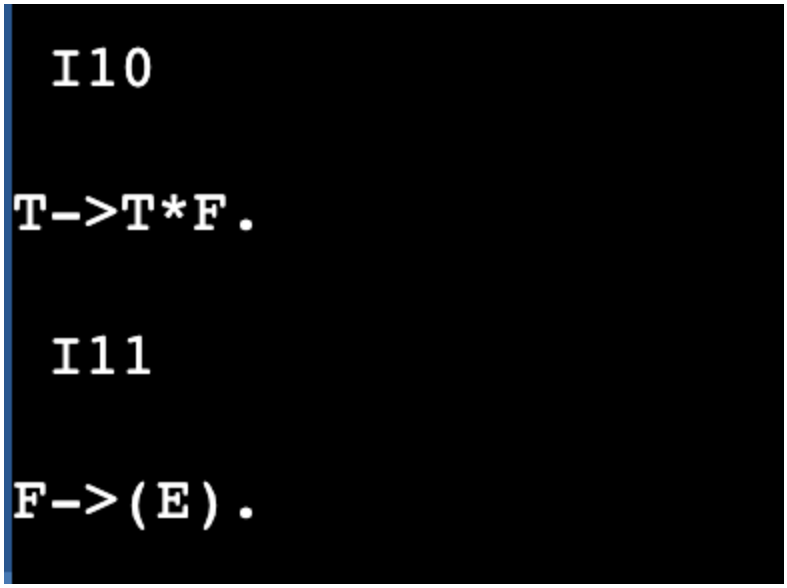
1. Start.
2. Create structure for production with LHS and RHS.
3. Open file and read input from file.
4. Build state 0 from extra grammar Law S' -> S $ that is all start symbol of grammar and one Dot (.) before S symbol.
5. If Dot symbol is before a non-terminal, add grammar laws that this non-terminal is in Left Hand Side of that Law and set Dot in before of first part of Right Hand Side.
6. If state exists (a state with this Laws and same Dot position), use that instead.
7. Now find set of terminals and non-terminals in which Dot exist in before.
8. If step 7 Set is non-empty go to 9, else go to 10.
9. For each terminal/non-terminal in set step 7 create new state by using all grammar law that Dot position is before of that terminal/non-terminal in reference state by increasing Dot point to next part in Right Hand Side of that laws.
10. Go to step 5.
11. End of state building.
12. Display the output.
13. End.

OUTPUT:

```
ENTER THE PRODUCTIONS OF THE GRAMMAR(0 TO END) :
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
0


 augumented grammar

A->E
E->E+T
E->T
T->T*F
T->F
F->(E)
F->i
 THE SET OF ITEMS ARE


 I0

A->.E
E->.E+T
E->.T
T->.T*F
T->.F
F->.(E)
F->.i

 I1

A->E.
E->E.+T

 I2

E->T.
T->T.*F
```

```
 I3

T->F.

 I4

F->(.E)
E->.E+T
E->.T
T->.T*F
T->.F
F->.(E)
F->.i

 I5

F->i.
```

```
 I6

E->E+.T
T->.T*F
T->.F
F->.(E)
F->.i

 I7

T->T*.F
F->.(E)
F->.i

 I8

F->(E.)
E->E.+T

 I9

E->E+T.
T->T.*F
```


I10
T->T*F.
I11
F->(E).