**Question No :01**

bexpr 🡪 bexpr or bterm | bterm

bterm 🡪 bterm and bfactor | bfactor

bfactor 🡪 not bfactor | (bexpr) | true | false

**Removing Left Recursion:**

Bexpr 🡪 bterm A’

A’ 🡪 or bterm | E

bterm 🡪 bfactor B’

B’ 🡪 and bfactor B’ | E

bfactor 🡪 not bfactor | (bexp) |

| true | false

**Finding First & Follow:**

**First:**

Bexpr = { not , ( , true , false }

A’ = or , E

Bterm = { not ,( , true , false }

B’ = [ and , E ]

bfactor = { not ; ( true , false )

**Follow:**

bex, pr 🡪 { $ , ) }

A’ 🡪 { $ , ‘ ) }

bterm 🡪 { or , ) , $ }

B’ 🡪 { or , ) , $ }

bfactor 🡪 { and , or , $ , ) }

**Parsing Table:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **or** | **and** | **(** | **)** | **true** | **false** | **not** | **$** |
| bexpr |  |  | 1 |  | 1 | 1 | 1 |  |
| bterm |  |  | 4 |  | 4 | 4 | 4 |  |
| bfactor |  |  | 8 |  | 9 | 10 | 7 |  |
| A’ | 2 |  |  | 3 |  |  |  | 3 |
| B’ | 6 | 5 |  | 6 |  |  |  | 6 |

**Question No :02**

**SLR IMPLEMENTATION**

A’ -> A

A -> A + K

A -> K

K-> KW

K-> W

W-> W+

W-> a

W-> b

STACK IMPLEMENTATION

|  |  |  |
| --- | --- | --- |
| **STACK** | **i/p** | **Production** |
| $bexpr | not(trueorfalse)$ | bexpr->btermA’ |
| $A’bterm | not(trueorfalse)$ | bterm->bfactorB’ |
| $A’B’bfactor | not(trueorfalse)$ | bfactor->not bfactor |
| $A’B’bfactornot | not(trueorfalse)$ |  |
| $A’B’bfactor | (trueorfalse)$ | bfactor->(bexp) |
| $A’B’)bexpr( | (trueorfalse)$ |  |
| $A’B’)bexpr | (trueorfalse)$ | bexp->btermA’ |
| $A’B’)A’B’bterm | (trueorfalse)$ | bterm->bfactorB’ |
| $A’B’)A’B’bfactor | (trueorfalse)$ | bfactor->true |
| $A’B’)A’B’true | (trueorfalse)$ |  |
| $A’B’)A’B’ | orfalse)$ | B’->Ɛ |
| $A’B’)A’ | orfalse)$ | A’->or bterm |
| $A’B’)bterm or | orfalse)$ |  |
| $A’B’)bterm | false)$ | bterm->bfactorB’ |
| $A’B’)Bbfactor | false)$ | Bfactor->false |
| $A’B’)B’false | false)$ |  |
| $A’B’)B’ | )$ | B’->Ɛ |
| $A’B’ | $ |  |
| $A’ | $ | B’->Ɛ |
| $ | $ | A’->Ɛ |

I9

I6

I1

I0

A-> A+K

K-> KW

W-> W\*

W-> a

W-> b

K

+

A

A’ -> A

A -> A + K

A-> K

K-> KW

K-> W

W-> W

W-> a

W-> b

A’ -> A

A -> A + K

A -> K

K-> KW

K-> W

W-> W

W-> a

W-> b

I3

W

I4

a

I2

I5

b

K

A-> K

K-> KW

W-> W

W-> a

W-> b

W

b

a

I5

I4

I3

W

I7

\*

K->KW

W->W\*

a

W

b

I8

K->W

W->W

W->b

a

I3

I4

I4

b

W->a

I5

I5

W->W\*

I7

**🡪**STATTIC IMPLEMENTATION

|  |  |  |
| --- | --- | --- |
| STACK | I/P | ACTION |
| 0 | A+ba\*+a\*$ | S4 |
| 0a4 | +ba\*+a\*$ | r6 W🡪E |
| 0W3 | +ba\*+a\*$ | r4 K🡪W |
| OK2 | +ba\*+a\*$ | r2 A🡪K |
| OA1 | +ba\*a\*$ | S6 |
| OA1+6 | ba\*a\*$ | S5 |
| OA1+6b5 | a\*+a\*$ | r7 W🡪b |
| OA1+6W3 | “” | r4 K🡪W |
| OA1+6K9 | “” | S4 |
| OA1+6K9a4 | \*+a\*$ | r6 W🡪a |
| OA1+6K9W3 | “” | S7 |
| OA1+6K9W3\*7 | +a\*$ | r5 W🡪W\* |
| OA1+6K9W8 | +a\*$ | r3 K🡪KW |
| OA1+6K9 | “” | r1 A🡪A+K |
| OA1 | “” | S6 |
| OA1+6 | a\*$ | S4 |
| OA1+6a4 | +$ | r6 W🡪a |
| OA1+6W3 | “” | S7 |
| OA1+6W3\*7 | $ | r5 W🡪W\* |
| OA1+6W3 | $ | r4 K🡪W |
| OA1+6K9 | $ | R1 A🡪A+K |
| OA1 | $ | acc |

Q3)

🡪S- Attribute STD:  
 If an STD uses only synthesized attribute , it is called S-ATTRIBUTE STD.

🡪L-ATTRIBUTE STD:  
 If an STD uses both synthesized attributes & inherited attributes with a restriction that inherited attributes can inherit values from left siblings only,it is called L-attribute STD.

**EXAMPLE:  
 P1:**S🡪MN{S.VAL = M.VAL+N.VAL}  
 **P2**:M🡪PQ{M.VAL=P.VAL\*Q.VAL & P.VAL=Q.VAL}

In P1,S is a synthesized attribute & in L-attribute definition synthesized is allowed, So, P1 follows the L-attributed definition ,but P2 doesn’t follow L-attributed definition as P is depending on Q which is RHS to it.