**COMP2022: Assignment 2**

**1.**

i) Variables of G = E, L, C, F, V, T

ii) Terminals of G = ( , ) , if , + , - , \* , 1 , 2 , 3 , 0 , a , b , c , d , print

iii) Start symbol of G = L

iv) L => E => (C) => (ifEE) => (if((F)E) => (if((-L)E) => (if((-LE)E) => (if((-EE)E) => (if((-TE)E) => (if((-1E)E) => (if((-1V)E) => (if((-1a)E) = > (if((-1a)(F)) => (if((-1a)(printL)) => (if((-1a)(printE)) => (if((-1a)(printT)) => (if((-1a)(print1))

v)



**2.**

LL(1) Parsers cannot parse non deterministic grammar, G is a non deterministic grammar. For the product C => ifEE | ifEEE, on seeing if, parser cannot decide whether to select the production rule C => ifEE or C => ifEEE. Therefore LL(1) parser fails to parse this grammar.

Apart from that, in the production of L => LE | E, there is a left recursion. LL(1) parsers cannot parse left recursive grammar, therefore the Grammar is not LL(1)

**3.**

To make the grammar G an LL(1) Grammar, We need to remove left-recursion and then we need to left factor the grammar.

Eliminating left recursion,

L => LE | E is the only product having left recursion.

L => EM

M=> EM |

Next we need to left factor the grammar,

E => (C) | (F) | V | T

C => ifEE | ifEEE

These are the products that we need to left factor

Product E => (C) | (F) | V | T contains the common prefix E = > ( and C => ifEE | ifEEE contains the common prefix C => ifEE.

Therefore, we can write E => (M | V | T

N => C) | F)

C => ifEED

D => | E

Final grammar G’ will be

L => EM

M => EM |

E => (N | V | T

N => C) | F)

C => ifEED

D => | E

F => +L | -L | \*L | printL

V => a | b | c | d

T => 0 | 1 | 2 | 3

**4.**

|  |  |  |
| --- | --- | --- |
| Product | First | Follow |
| L => EM | { ( , a , b , c, d , 0 , 1 , 2 , 3 } | { $ , ) } |
| M=> EM | | { ( , a , b , c, d , 0 , 1 , 2 , 3, } | { $, ) } |
| E => N | V | T | { ( , a , b , c , d , 0 , 1 , 2 , 3 } | { ( , a , b , c, d , 0 , 1 , 2 , 3, $ , ) } |
| N => C) | F) | { if, + , - , \* , print } | { ( , a , b , c, d , 0 , 1 , 2 , 3, $ , ) } |
| C => ifEED | { if } | { ) } |
| D => | E | { ( , a , b , c, d , 0 , 1 , 2 , 3, } | { ) } |
| F => +L | -L | \*L | printL | { + , - , \* , print } | { ) } |
| V => a | b | c | d | { a , b , c , d } | { ( , a , b , c, d , 0 , 1 , 2 , 3, $ , ) } |
| T => 0 | 1 | 2 | 3 | { 0 , 1 , 2 , 3 } | { ( , a , b , c, d , 0 , 1 , 2 , 3, $ , ) } |

Parser Table...

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | ( | ) | if | + | - | \* | 1 | 2 | 3 | 0 | a | b | c | d | print | $ |
| L | L => EM |  |  |  |  |  | L => EM | L => EM | L => EM | L => EM | L => EM | L => EM | L => EM | L => EM |  |  |
| M | M => EM | M => |  |  |  |  | M => EM | M => EM | M => EM | M => EM | M => EM | M => EM | M => EM | M => EM |  | M => |
| E | E => (N |  |  |  |  |  | E => T | E => T | E => T | E => T | E => V | E => V | E => V | E => V |  |  |
| N |  |  | N => C) | N => F) | N => F) | N => F) |  |  |  |  |  |  |  |  | N => F) |  |
| C |  |  | C => ifEED |  |  |  |  |  |  |  |  |  |  |  |  |  |
| D | D => E | D => |  |  |  |  | D => E | D => E | D => E | D => E | D => E | D => E | D => E | D => E |  | D => |
| F |  |  |  | F => +L | F => -L | F => \*L |  |  |  |  |  |  |  |  | F => printL |  |
| V |  |  |  |  |  |  |  |  |  |  | V => a | V => b | V => c | V => d |  |  |
| T |  |  |  |  |  |  | T => 1 | T => 2 | T => 3 | T => 0 |  |  |  |  |  |  |