Sparsh Agarwal

CS-536

9075905142

HW 4

A1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | - | - | - | - |
|  |  | - | - | - |
|  | S |  | - | - |
| F | B | A | F | - |
| D | D | E | D | D |
| a | a | b | a | a |

Since only S is possible in the third row and we cannot move to the top for entire sequence to be generated, we can only get “aba” from this grammar according to CYK algorithm

A2. Assuming, discrete values are returned

program → MAIN LPAREN RPAREN LCURLY list RCURLY

program.trans = program.trans U list.trans  
  
list → list oneItem list.trans = list.trans U list2.trans U oneItem.trans  
 | epsilon list.trans = list.trans U {}  
   
oneItem → decl no translation necessary  
 | stmt oneItem.trans = oneItem.trans U stmt.trans  
   
decl → BOOL ID SEMICOLON no translation necessary  
 | INT ID SEMICOLON no translation necessary  
   
stmt → ID ASSIGN exp SEMICOLON stmt.trans = stmt.trans U exp.trans  
 | IF LPAREN exp RPAREN stmt stmt.trans = stmt.trans U exp.trans U stmt2.trans  
 | WHILE LPAREN exp RPAREN stmt stmt.trans = stmt.trans U exp.trans U stmt2.trans  
 | LCURLY list RCURLY stmt.trans = stmt.trans U list.trans  
  
exp → exp TIMES exp exp.trans= exp.trans U exp2.trans U exp3.trans  
 | exp DIVIDE exp exp.trans= exp.trans U exp2.trans U exp3.trans  
 | exp PLUS exp exp.trans= exp.trans U exp2.trans U exp3.trans  
 | exp LESS exp exp.trans= exp.trans U exp2.trans U exp3.trans  
 | exp EQUALS exp exp.trans= exp.trans U exp2.trans U exp3.trans  
 | LPAREN exp RPAREN exp.trans= exp.trans U exp2.trans

| ID no translation necessary  
 | BOOLLITERAL exp.trans= exp.trans U {}  
 | INTLITERAL exp.trans= exp.trans U { INTLITERAL.value }