

Grammar:
 0. start ::= stmt
 1. stmt ::= "print" exp
 2. exp ::= exp "+" exp
 3. exp ::= INT

State 0

start ::= . stmt
 stmt ::= . "print" exp

State 1

stmt ::= "print" . exp
 exp ::= . exp "+" exp
 exp ::= . INT

"print", shift

stmt, goto

INT, shift

exp, goto

State 2

start ::= stmt .

State 4

exp ::= INT .
 end, reduce by rule 3
 "+", reduce by rule 3

State 3

stmt ::= "print" exp .
 exp ::= exp . "+" exp
 end, reduce by rule 1

State 6

exp ::= exp "+" exp .
 exp ::= exp . "+" exp
 end, reduce by rule 2
 "+", reduce by rule 2

INT, shift

"+", shift

State 5

exp ::= exp "+" . exp
 exp ::= . exp "+" exp
 exp ::= . INT

"+", shift

exp, goto

Example parse of 'print 1 + 2'

Stack	Input	Action
[]	'print 1 + 2'	shift to state 1
[(1,"print")]	'1 + 2'	shift to state 4
[(1,"print"),(4,INT)]	'+ 2'	reduce by rule 3 to state 1, goto 3
[(1,"print"),(3,exp)]	'+ 2'	shift to state 5
[(1,"print"),(3,exp),(5,+)]	'2'	shift to state 4
[(1,"print"),(3,exp),(5,+),(4,INT)]	"	reduce by rule 3 to state 5, goto 6
[(1,"print"),(3,exp),(5,+),(6,exp)]	"	reduce by rule 2 to state 1, goto 3
[(1,"print"),(3,exp)]	"	reduce by rule 1 to state 0, goto 2
[(2,stmt)]	"	accept