step	Stack	Input	Action
1	$0	(id+id)*id$	shift 4 (s4)
2	$0(4	id+id)*id$	shift 5 (s5)
3	$0(4 id5	+id)*id$	reduce F→id (r6)
4	$0(4 F3	+id)*id$	reduce T→F (r4)
5	$0(4 T2	+id)*id$	reduce E→T (r2)
6	$0(4 E8	+id)*id$	shift 6 (s6)
7	$0(4E8+6	id)*id$	shift 5 (s5)
8	$0(4E8+6 id5	)*id$	reduce F→id(r6)
9	$0(4E8+6 F3	)*id$	reduce T→F (r4)
10	$0(4E8+6T9	)*id$	reduce E→E+T(r1)
11	$0(4 E8	)*id$	shift 11 (s11)
12	$0(4E8)11	*id$	reduce F→(E)(r5)
13	$0 F3	*id$	reduce T→F (r4)
14	$0 T2	*id$	shift 7 (s7)
15	$0T2*7	id$	shift 5 (s5)
16	$0T2*7id5	$	reduce F→id(r6)
17	$0T2*7F10	$	reduce T→T*F(r3)
18	$0T2	$	reduce E→T(r2)
19	$0E1	$	accept

parse tree (id+id)*id$
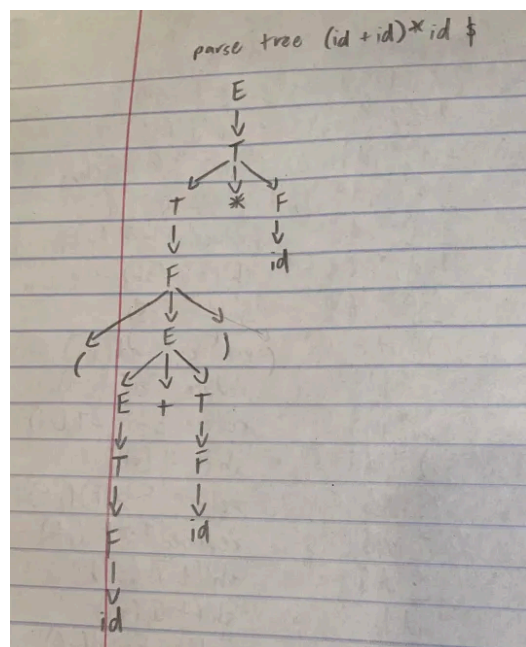
INPUT 1: "(id+id)*id$"

INPUT 2: "id*id$"

1	$0	id*id$	shift 5 (s5)
2	$0 id5	*id$	reduce F→id (r6)
3	$0 F3	*id$	reduce T→F (r4)
4	$0 T2	*id$	shift 7 (s7)
5	$0 T2 *7	id$	shift 5 (s5)
6	$0 T2 *7 id5	$	reduce F→id (r6)
7	$0 T2 *7 F10	$	reduce T→T*F (r3)
8	$0 T2	$	reduce E→T (r2)
9	$0 E1.	$	accept

```
E
↓
T
↙ ↓ ↘
T  *  F
↓     ↓
F    id
↓
id
```

parse tree
id * id $

INPUT 3: "(id*)$"

	stack	Input	Action
1	$0	(id*)$	shift 4 (s4)
2	$0 (4	id*)$	shift 5 (s5)
3	$0(4 id5	*)$	reduce F→id(r6)
4	$0(4 F3	*)$	reduce T→F(r4)
5	$0(4 T2	*)$	shift 7 (s7)
6	$0(4 T2 *7	)$	error

parse tree   error
(id*)$

```
(   T   *
    ↓
    F
    ↓
    id
```