

# تکلیف دوم

مجتبی ملائی  
۴۰۱۳۱۳۸۳

۱

(آ) با بررسی در  $S$  می‌بینیم که  $S \xRightarrow{+} S\alpha$  وجود دارد. ابتدا در  $S \rightarrow AbS$  قانون  $A$  را با سمت راست آن جایگزین می‌کنیم:

$$\begin{aligned} S &\rightarrow SaS \mid SaAbS \mid BbS \\ A &\rightarrow SaA \mid B \\ B &\rightarrow bS \mid c \end{aligned}$$

سپس طبق قانون گفته شده در اسلاید داریم:

$$\begin{aligned} S &\rightarrow BbSS' \\ S' &\rightarrow aSS' \mid aAbSS' \mid \epsilon \\ A &\rightarrow SaA \mid B \\ B &\rightarrow bS \mid c \end{aligned}$$

(ب) برای ساده سازی ابتدا  $B$  و  $C$  را جایگزین می‌کنیم.

$$\begin{aligned} S &\rightarrow abcA \mid abcb \mid abc \\ A &\rightarrow abA \mid abbA \mid abbc \end{aligned}$$

سپس  $abc$  را در  $S$  فاکتور می‌گیریم.

$$\begin{aligned} S &\rightarrow abcS' \\ S' &\rightarrow A \mid b \mid \epsilon \\ A &\rightarrow abA \mid abbA \mid abbc \end{aligned}$$

در  $A$  حروف  $ab$  را فاکتور می‌گیریم.

$$\begin{aligned} S &\rightarrow abcS' \\ S' &\rightarrow A \mid b \mid \epsilon \\ A &\rightarrow abA' \\ A' &\rightarrow A \mid bA \mid bc \end{aligned}$$

حال می‌توانیم دوباره  $b$  را فاکتور بگیریم.

$$\begin{aligned} S &\rightarrow abcS' \\ S' &\rightarrow A \mid b \mid \epsilon \\ A &\rightarrow abA' \\ A' &\rightarrow A \mid bA'' \\ A'' &\rightarrow A \mid c \end{aligned}$$

Non-terminals	First	Follow
Program	{	\$
Statements	id, if, $\epsilon$	}
Statement	id, if	id, if, }
Expression	id	;, )
Tail	+, -, $\epsilon$	;, )

Table 1: First and Follow sets for non-terminals

```

1 // Assume we have a function `nextToken()` that gets the next token from the input stream
2 // Assume `currentToken` holds the current token
3 // Assume `match(expected)` matches the current token and advances to the next one
4 function Program():
5     if currentToken == '{':
6         match('{')
7         Statements()
8         match('}')
9         match('eof') // Ensure the program ends correctly
10    else:
11        error("Expected '{' at the start of the program")
12
13 function Statements():
14     if currentToken in {'id', 'if'}: // FIRST(Statements)
15         Statement()
16         Statements()
17     else:
18         // epsilon (FOLLOW(Statements) is { '}' }), so we return without consuming anything
19 function Statement():
20     if currentToken == 'id':
21         match('id')
22         match('=')
23         Expression()
24         match(';')
25     else if currentToken == 'if':
26         match('if')
27         match('(')
28         Expression()
29         match(')')
30         Statement()
31     else:
32         error("Invalid statement")
33
34 function Expression():
35     if currentToken == 'id':
36         match('id')
37         Tail()
38     else:
39         error("Expected identifier in expression")
40
41 function Tail():
42     if currentToken == '+':
43         match('+')
44         Expression()
45     else if currentToken == '-':
46         match('-')
47         Expression()
48     else:
49         // epsilon (FOLLOW(Tail) is { ';', '}' }), so we return without consuming anything

```

Listing 1: Recursive Descent Parser Pseudo-Code

.۱

Non-terminals	First	Follow
S	if	\$
I	=, $\epsilon$	\$
E	(, id, num	\$, ), then
E'	+, $\epsilon$	\$, ), then
T	(, id, num	+, \$, ), then
T'	*, $\epsilon$	+, \$, ), then
F	(, id, num	*, +, \$, ), then

.۲

	<i>id</i>	<i>num</i>	(	)	+	*	=	<i>if</i>	<i>then</i>	\$
<i>S</i>	<i>idI</i>							<i>ifEthenS</i>		
<i>I</i>										<i>e</i>
<i>E</i>	<i>TE'</i>	<i>TE'</i>	<i>TE'</i>							
<i>E'</i>				$\epsilon$	<i>+TE'</i>				$\epsilon$	$\epsilon$
<i>T</i>	<i>FT'</i>	<i>FT'</i>	<i>FT'</i>							
<i>T'</i>				$\epsilon$	$\epsilon$	<i>*FT'</i>			$\epsilon$	$\epsilon$
<i>F</i>	<i>id</i>	<i>num</i>	( <i>E</i> )							

.۳

Step	Matched	Stack	Input	Action
1	$\epsilon$	$S$	$if\ id\ then\ id = (num * id) + num\ \$$	
2	$\epsilon$	$if\ E\ then\ S$	$if\ id\ then\ id = (num * id) + num\ \$$	output $S \rightarrow if\ E\ then\ S$
3	$if$	$E\ then\ S$	$id\ then\ id = (num * id) + num\ \$$	match $if$
4	$if$	$T\ E'\ then\ S$	$id\ then\ id = (num * id) + num\ \$$	output $E \rightarrow T\ E'$
5	$if$	$F\ T'\ E'\ then\ S$	$id\ then\ id = (num * id) + num\ \$$	output $T \rightarrow F\ T'$
6	$if$	$id\ T'\ E'\ then\ S$	$id\ then\ id = (num * id) + num\ \$$	output $F \rightarrow id$
7	$if\ id$	$T'\ E'\ then\ S$	$then\ id = (num * id) + num\ \$$	match $id$
8	$if\ id$	$E'\ then\ S$	$then\ id = (num * id) + num\ \$$	output $T' \rightarrow \epsilon$
9	$if\ id$	$then\ S$	$then\ id = (num * id) + num\ \$$	output $E' \rightarrow \epsilon$
10	$if\ id\ then$	$S$	$id = (num * id) + num\ \$$	match $then$
11	$if\ id\ then$	$id\ I$	$id = (num * id) + num\ \$$	output $S \rightarrow id\ I$
12	$if\ id\ then\ id$	$I$	$= (num * id) + num\ \$$	match $id$
13	$if\ id\ then\ id$	$=\ E$	$= (num * id) + num\ \$$	output $I \rightarrow =\ E$
14	$if\ id\ then\ id =$	$E$	$(num * id) + num\ \$$	match $=$
15	$if\ id\ then\ id =$	$T\ E'$	$(num * id) + num\ \$$	output $E \rightarrow T\ E'$
16	$if\ id\ then\ id =$	$F\ T'\ E'$	$(num * id) + num\ \$$	output $T \rightarrow F\ T'$
17	$if\ id\ then\ id =$	$(\ E)\ T'\ E'$	$(num * id) + num\ \$$	output $F \rightarrow (E)$
18	$if\ id\ then\ id = ($	$E)\ T'\ E'$	$num * id) + num\ \$$	match $($
19	$if\ id\ then\ id = ($	$T\ E')\ T'\ E'$	$num * id) + num\ \$$	output $E \rightarrow T\ E'$
20	$if\ id\ then\ id = ($	$F\ T'\ E')\ T'\ E'$	$num * id) + num\ \$$	output $T \rightarrow F\ T'$
21	$if\ id\ then\ id = ($	$num\ T'\ E')\ T'\ E'$	$num * id) + num\ \$$	output $F \rightarrow num$
22	$if\ id\ then\ id = (num$	$T'\ E')\ T'\ E'$	$*id) + num\ \$$	match $num$
23	$if\ id\ then\ id = (num$	$*\ F\ T'\ E')\ T'\ E'$	$id) + num\ \$$	output $T' \rightarrow *\ F\ T'$
24	$if\ id\ then\ id = (num *$	$F\ T'\ E')\ T'\ E'$	$id) + num\ \$$	match $*$
25	$if\ id\ then\ id = (num *$	$id\ T'\ E')\ T'\ E'$	$id) + num\ \$$	output $F \rightarrow id$
26	$if\ id\ then\ id = (num * id$	$T'\ E')\ T'\ E'$	$) + num\ \$$	match $id$
27	$if\ id\ then\ id = (num * id$	$E')\ T'\ E'$	$) + num\ \$$	output $T' \rightarrow \epsilon$
28	$if\ id\ then\ id = (num * id$	$)\ T'\ E'$	$) + num\ \$$	output $E' \rightarrow \epsilon$
29	$if\ id\ then\ id = (num * id)$	$T'\ E'$	$+ num\ \$$	match $)$
30	$if\ id\ then\ id = (num * id)$	$+ T\ E'$	$num\ \$$	output $E' \rightarrow +T\ E'$
31	$if\ id\ then\ id = (num * id) +$	$T\ E'$	$num\ \$$	match $+$
32	$if\ id\ then\ id = (num * id) +$	$F\ T'\ E'$	$num\ \$$	output $T \rightarrow F\ T'$
33	$if\ id\ then\ id = (num * id) +$	$num\ T'\ E'$	$num\ \$$	output $F \rightarrow num$
34	$if\ id\ then\ id = (num * id) + num$	$T'\ E'$	$\$$	match $num$
35	$if\ id\ then\ id = (num * id) + num$	$E'$	$\$$	output $T' \rightarrow \epsilon$
36	$if\ id\ then\ id = (num * id) + num$		$\$$	output $E' \rightarrow \epsilon$
37	$if\ id\ then\ id = (num * id) + num\ \$$			match $\$$

Table 2: LL(1) Parsing Trace for  $if\ id\ then\ id = (num * id) + num$