22100409 Siaw Jia Yuin

LL1 Parser

BNF

S -> BEGIN_BLOCK BEGIN_STMT

BEGIN_BLOCK -> program PROGRAM_NAME

PROGRAM_NAME -> id

BEGIN_STMT -> begin STMT* end

STMT -> DECLARATION
| IF_STMT
| PRINT_STMT
| WHILE_STMT
| FOR_STMT
| BREAK_STMT

DECLARATION -> int OPERATOR ;
| OPERATOR ,

OPERATOR -> id = OP_STMT
| OP_STMT

OP_STMT -> TERM
| OP_STMT + TERM
| id ++

TERM -> TERM * FACTOR
| FACTOR

FACTOR -> number_literal
| id

PRINT_CONTENT -> string_literal
| id

IF_STMT -> if (COM_STMT) BEGIN_STMT
| if (COM_STMT) BEGIN_STMT else_if (OP_STMT) BEGIN_STMT
| if (COM_STMT) BEGIN_STMT else_if (OP_STMT) BEGIN_STMT else BEGIN_STMT

PRINT_STMT -> print_line (PRINT_CONTENT) ;
| display (PRINT_CONTENT) ;

WHILE_STMT -> while (STMT) BEGIN_STMT

FOR_STMT -> for (STMT) BEGIN_STMT

BREAK_STMT -> break ;

COM_STMT -> LT_STMT
| EQ_STMT

LT_STMT -> FACTOR < FACTOR

EQ_STMT -> FACTOR == FACTOR

BNF without Left Recursion

S -> BEGIN_BLOCK BEGIN_STMT

BEGIN_BLOCK -> program id

BEGIN_STMT -> begin STMTS end

STMTS -> STMT STMTS
STMTS -> ''

STMT -> DECLARATION_LIST
STMT -> IF_STMT
STMT -> PRINT_STMT
STMT -> WHILE_STMT
STMT -> FOR_STMT
STMT -> BREAK_STMT

DECLARATION_LIST -> DECLARATION DECLARATION_LIST'
DECLARATION_LIST -> ‘’

DECLARATION_LIST' -> , DECLARATION DECLARATION_LIST’
DECLARATION_LIST' -> ''

DECLARATION -> TYPE ID_LIST ;
DECLARATION -> ID_LIST ;

TYPE -> int
TYPE -> integer

ASSIGNMENT -> = OP_STMT
ASSIGNMENT -> ‘’

ID_LIST -> id ASSIGNMENT ID_LIST'
ID_LIST' -> , id ASSIGNMENT ID_LIST'
ID_LIST' -> ''

OP_STMT -> TERM OP_STMT'

OP_STMT' -> + TERM OP_STMT'
OP_STMT’ -> ++
OP_STMT' -> ''

TERM -> FACTOR TERM'

TERM' -> * FACTOR TERM'
TERM' -> ''

FACTOR -> number_literal
FACTOR -> id

PRINT_CONTENT -> string_literal
PRINT_CONTENT -> identifier

IF_STMT -> if (COM_STMT) BEGIN_STMT IF_STMT'

IF_STMT' -> else_if (COM_STMT) BEGIN_STMT IF_STMT'
IF_STMT' -> else BEGIN_STMT
IF_STMT' -> ''

PRINT_STMT -> print_line (PRINT_CONTENT) ;
PRINT_STMT -> display (PRINT_CONTENT) ;

WHILE_STMT -> while (COM_STMT) BEGIN_STMT

FOR_STMT -> for (DECLARATION COM_STMT ; OP_STMT) BEGIN_STMT

BREAK_STMT -> break ;

COM_STMT -> TERM COM_STMT’

COM_STMT’ -> < FACTOR
COM_STMT’ -> == FACTOR
COM_STMT’ -> ‘’

First Set

S = {program}

BEGIN_BLOCK = {program}

BEGIN_STMT = {begin}

STMTS = { /eps, if, print_line, display, while, for, break, int, integer, id }

STMT = { /eps, if, print_line, display, while, for, break, int, integer, id }

DECLARATION_LIST = { /eps, int, integer, id }

DECLARATION_LIST' = { /eps , , }

DECLARATION = { int, integer, id }

TYPE = { int, integer }

ASSIGNMENT = { =, /eps }

ID_LIST = {id}

ID_LIST' = { , , /eps }

OP_STMT = { number_literal, id }

OP_STMT' = { +, ++, /eps }

TERM = { number_literal, id }

TERM' = { *, /eps }

FACTOR = { number_literal, id }

PRINT_CONTENT = { string_literal, id }

IF_STMT = { if }

IF_STMT' = { else_if, else, /eps }

PRINT_STMT = { print_line, display }

WHILE_STMT = { while }

FOR_STMT = { for }

BREAK_STMT = { break }

COM_STMT = { number_literal, id }

COM_STMT' = { <, ==, /eps }

Follow Set

S = { $ }

BEGIN_BLOCK = { begin }

BEGIN_STMT = { $, else_if, else, end, if, print_line, display, while, for, break, int, integer, id }

STMTS = { end }

STMT = { end, if, print_line, display, while, for, break, int, integer, id }

DECLARATION_LIST = { end, if, print_line, display, while, for, break, int, integer, id }

DECLARATION_LIST' = { end, if, print_line, display, while, for, break, int, integer, id }

DECLARATION = { end, if, print_line, display, while, for, break, int, integer, id, , ,DECLARATION_LIST', number_Literal }

TYPE = { id }

ASSIGNMENT = { , , ; }

ID_LIST = {;}

ID_LIST' = { ; }

OP_STMT = {), , , ; }

OP_STMT' = {), , , ; }

TERM = { +, ++, <, ==,), ;, , }

TERM' = { +, ++, <, ==,), ;, , }

FACTOR = { *, +, ++,), ;, <, ==, , }

PRINT_CONTENT = {) }

IF_STMT = { end, if, print_line, display, while, for, break, int, integer, id }

IF_STMT' = { end, if, print_line, display, while, for, break, int, integer, id }

PRINT_STMT = { end, if, print_line, display, while, for, break, int, integer, id }

WHILE_STMT = { end, if, print_line, display, while, for, break, int, integer, id }

FOR_STMT = { end, if, print_line, display, while, for, break, int, integer, id }

BREAK_STMT = { end, if, print_line, display, while, for, break, int, integer, id }

COM_STMT = {), ; }

COM_STMT' = {), ; }

Parsing Table

Nonterminal	program	id	begin	end	,	DECLARATION_LIST'	;	int	integer	=	+	++	*	number_literal	string_literal	if	(	)	else_if	else	print_line	display	while	for	break	<	==	$
S	S -> BEGIN_BLOCK BEGIN_STMT																											
BEGIN_BLOCK	BEGIN_BLOCK -> program id																											
BEGIN_STMT			BEGIN_STMT -> begin STMTS end																									
STMTS		STMTS -> STMT STMTS STMT -> DECLARATION_LIST		STMTS -> STMT STMTS																								
STMT																												
DECLARATION_LIST		DECLARATION_LIST -> ;	DECLARATION_LIST -> ' '		DECLARATION_LIST -> , DECLARATION DECLARATION_LIST			DECLARATION_LIST -> ' '	DECLARATION_LIST -> ' '							DECLARATION_LIST -> ' '				DECLARATION_LIST -> ' '	DECLARATION_LIST -> ' '	DECLARATION_LIST -> ' '	DECLARATION_LIST -> ' '	DECLARATION_LIST -> ' '	DECLARATION_LIST -> ' '		
DECLARATION		DECLARATION -> ID_LIST ;						DECLARATION -> TYPE ID_LIST ;	DECLARATION -> TYPE ID_LIST ;																		
TYPE								TYPE -> int	TYPE -> integer																		
ASSIGNMENT					ASSIGNMENT -> ' '		ASSIGNMENT -> ' '			ASSIGNMENT -> = OP_STMT																
ID_LIST		ID_LIST -> id ASSIGNMENT ID_LIST'																									
ID_LIST'					ID_LIST' -> , id ASSIGNMENT ID_LIST'		ID_LIST' -> ' '																				
OP_STMT		OP_STMT -> TERM OP_STMT'												OP_STMT -> TERM OP_STMT'													
OP_STMT'					OP_STMT' -> ' '		OP_STMT' -> ' '				OP_STMT' -> + TERM OP_STMT'	OP_STMT' -> ++							OP_STMT' -> ' '									
TERM		TERM -> FACTOR TERM'												TERM' -> FACTOR TERM'													
TERM'					TERM' -> ' '		TERM' -> ' '				TERM' -> *		TERM' -> ' ' FACTOR TERM'					TERM' -> ' '								TERM' -> ' '	TERM' -> ' '
FACTOR		FACTOR -> id												FACTOR -> number_literal													
PRINT_CONTENT		PRINT_CONTENT -> id													PRINT_CONTENT -> string_literal												
IF_STMT																IF_STMT -> if (COM_STMT) BEGIN_STMT IF_STMT'											
IF_STMT'		IF_STMT -> ' '		IF_STMT' -> ' '				IF_STMT' -> ' '	IF_STMT' -> ' '							IF_STMT' -> ' '			IF_STMT' -> else_if (COM_STMT) BEGIN_STMT IF_STMT	IF_STMT' -> else BEGIN_STMT	IF_STMT' -> ' '	IF_STMT' -> ' '	IF_STMT' -> ' '	IF_STMT' -> ' '	IF_STMT' -> ' '		
PRINT_STMT																					PRINT_STMT -> print_line (PRINT_CONTENT) ;	PRINT_STMT -> display (PRINT_CONTENT) ;					
WHILE_STMT																							WHILE_STMT -> while (COM_STMT) BEGIN_STMT				
FOR_STMT																								FOR_STMT -> for (DECLARATION COM_STMT ; OP_STMT) BEGIN_STMT			
BREAK_STMT																									BREAK_STMT -> break ;		
COM_STMT		COM_STMT -> TERM COM_STMT'												COM_STMT -> TERM COM_STMT'													
COM_STMT'							COM_STMT' -> ' '											COM_STMT' -> ' '								COM_STMT' -> < FACTOR	COM_STMT' -> == FACTOR	

Program Output

1. Test1

```
[GENERATE-stack] [$, BEGIN_STMT, BEGIN_BLOCK]
[GENERATE-stack] [$, BEGIN_STMT, identifier, program]
[MATCH] - KEYWORD - program
[MATCH] - IDENTIFIER - Test1
[GENERATE-stack] [$, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, TYPE]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, int]
[MATCH] - KEYWORD - int
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - Time.10.24
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 2206
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - Hour.In.Day
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 0
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;]
```

```
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT, (, if]
[MATCH] - KEYWORD - if
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - Time.10.24
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), FACTOR, <]
[MATCH] - LESS THAN OPERATOR - <
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), number_literal]
[MATCH] - NUMBER_LITERAL - 10
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, print_line]
[MATCH] - KEYWORD - print_line
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "Good morning."
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - Hour.In.Day
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
```

```
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 24
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT, (, else_if]
[MATCH] - KEYWORD - else_if
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - Time.10.24
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), FACTOR, <]
[MATCH] - LESS THAN OPERATOR - <
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), number_literal]
[MATCH] - NUMBER_LITERAL - 20
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - Hour.In.Day
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR, *]
[MATCH] - MULTIPLICATION OPERATOR - *
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 60
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
```

```
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, print_line]
[MATCH] - KEYWORD - print_line
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "Good day."
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, else]
[MATCH] - KEYWORD - else
[GENERATE-stack] [$, end, STMTS_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, print_line]
[MATCH] - KEYWORD - print_line
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "Good evening."
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end]
[MATCH] - KEYWORD - end
Parse OK
```

2. Test1error

```
[GENERATE-stack] [$, BEGIN_STMT, BEGIN_BLOCK]
[GENERATE-stack] [$, BEGIN_STMT, identifier, program]
[MATCH] - KEYWORD - program
[MATCH] - IDENTIFIER - Testlerror
[GENERATE-stack] [$, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, TYPE]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, int]
[MATCH] - KEYWORD - int
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - Time.10.24
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 2206
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - Hour.In.Day
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 0
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
```

```
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT, (, if]
[MATCH] - KEYWORD - if
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - Time.10.24
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), FACTOR, <]
[MATCH] - LESS THAN OPERATOR - <
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), number_literal]
[MATCH] - NUMBER_LITERAL - 10
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, print_line]
[MATCH] - KEYWORD - print_line
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "Good morning."
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
```

```
[MATCH] - IDENTIFIER - Hour.In.Day
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 24
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT, (, else_if]
[MATCH] - KEYWORD - else_if
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - Time.10.24
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), FACTOR, <]
[MATCH] - LESS THAN OPERATOR - <
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), number_literal]
[MATCH] - NUMBER_LITERAL - 20
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - Hour.In.Day
```

```
[MATCH] - IDENTIFIER - Hour.In.Day
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR, *]
[MATCH] - MULTIPLICATION OPERATOR - *
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 60
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, print_line]
[MATCH] - KEYWORD - print_line
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "Good day."
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, IF_STMT_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, else]
[MATCH] - KEYWORD - else
[GENERATE-stack] [$, end, STMTS_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, print_line]
[MATCH] - KEYWORD - print_line
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "Good evening."
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end]
[MATCH] - KEYWORD - end
ERROR: Parsing stack is empty.
Keyword end not matched.
Parsing failed.
```

3. Test2

```
[GENERATE-stack] [$, BEGIN_STMT, BEGIN_BLOCK]
[GENERATE-stack] [$, BEGIN_STMT, identifier, program]
[MATCH] - KEYWORD - program
[MATCH] - IDENTIFIER - Test2
[GENERATE-stack] [$, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, TYPE]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, int]
[MATCH] - KEYWORD - int
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - i.Value.$
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 8
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 1
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;]
```

```
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, WHILE_STMT]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT, (, while]
[MATCH] - KEYWORD - while
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - i.Value.$
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), FACTOR, <]
[MATCH] - LESS THAN OPERATOR - <
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), number_literal]
[MATCH] - NUMBER_LITERAL - 5
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, print_line]
[MATCH] - KEYWORD - print_line
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), identifier]
[MATCH] - IDENTIFIER - i.Value.$
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - i.Value.$
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
```

```
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - i.Value.$
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM, +]
[MATCH] - PLUS OPERATOR - +
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 1
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM, +]
[MATCH] - PLUS OPERATOR - +
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 10
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end]
[MATCH] - KEYWORD - end
Parse OK
```

4. Test2error

```
[GENERATE-stack] [$, BEGIN_STMT, BEGIN_BLOCK]
[GENERATE-stack] [$, BEGIN_STMT, identifier, program]
[MATCH] - KEYWORD - program
[MATCH] - IDENTIFIER - Test2error
[GENERATE-stack] [$, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, TYPE]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, int]
[MATCH] - KEYWORD - int
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - i.Value.$
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 0
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - $Minute.In.Hour
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 1
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
ERROR: keyword 'while4' spelling error.
Parsing failed.
```

5. Test3

```
[GENERATE-stack] [$, BEGIN_STMT, BEGIN_BLOCK]
[GENERATE-stack] [$, BEGIN_STMT, identifier, program]
[MATCH] - KEYWORD - program
[MATCH] - IDENTIFIER - Test3
[GENERATE-stack] [$, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, TYPE]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, integer]
[MATCH] - KEYWORD - integer
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - Time_10_24
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 2206
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - Hour_In_Day
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 0
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - temp
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 0
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
```

```
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - Minute_In_Hour
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 3
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, FOR_STMT]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, DECLARATION, (, for]
[MATCH] - KEYWORD - for
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST, TYPE]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST, int]
[MATCH] - KEYWORD - int
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - i
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 0
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - i
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, FACTOR, <]
[MATCH] - LESS THAN OPERATOR - <
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), number_literal]
[MATCH] - NUMBER_LITERAL - 10
```

```
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - i
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), ++]
[MATCH] - INCREMENT OPERATOR - ++
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, display]
[MATCH] - KEYWORD - display
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "test input"
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, WHILE_STMT]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT, (, while]
[MATCH] - KEYWORD - while
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - temp
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), FACTOR, <]
[MATCH] - LESS THAN OPERATOR - <
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), number_literal]
[MATCH] - NUMBER_LITERAL - 10
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST]
```

```
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - temp
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - temp
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM, +]
[MATCH] - PLUS OPERATOR - +
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 1
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, display]
[MATCH] - KEYWORD - display
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "Increase temperature"
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT, (, if]
[MATCH] - KEYWORD - if
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - temp
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), FACTOR, ==]
[MATCH] - EQUALS OPERATOR - ==
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, BEGIN_STMT, ), number_literal]
[MATCH] - NUMBER_LITERAL - 5
```

```
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, display]
[MATCH] - KEYWORD - display
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "stop increasing temp"
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS_P, BREAK_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end, STMTS_P, ;, break]
[MATCH] - KEYWORD - break
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, IF_STMT_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end]
[MATCH] - KEYWORD - end
Parse OK
```

6. Test3error

```
[GENERATE-stack] [$, BEGIN_STMT, BEGIN_BLOCK]
[GENERATE-stack] [$, BEGIN_STMT, identifier, program]
[MATCH] - KEYWORD - program
[MATCH] - IDENTIFIER - Test3Error
[GENERATE-stack] [$, end, STMTS, begin]
[MATCH] - KEYWORD - begin
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, DECLARATION]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, TYPE]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST, integer]
[MATCH] - KEYWORD - integer
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - Time_10_24
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 2206
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - Hour_In_Day
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 0
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, ASSIGNMENT, identifier, ,]
[MATCH] - PUNCTUATION COMMA - ,
[MATCH] - IDENTIFIER - Minute_In_Hour
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 3
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;, ID_LIST_P]
```

```
[GENERATE-stack] [$, end, STMTS_P, DECLARATION_LIST_P, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P]
[GENERATE-stack] [$, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, FOR_STMT]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, DECLARATION, (, for]
[MATCH] - KEYWORD - for
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST, TYPE]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST, int]
[MATCH] - KEYWORD - int
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, ASSIGNMENT, identifier]
[MATCH] - IDENTIFIER - i
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT, =]
[MATCH] - ASSIGNMENT OPERATOR - =
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT_P, TERM_P, number_literal]
[MATCH] - NUMBER_LITERAL - 0
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P, OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;, ID_LIST_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT, ;]
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - i
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, COM_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, FACTOR, <]
[MATCH] - LESS THAN OPERATOR - <
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT, ;, number_literal]
[MATCH] - NUMBER_LITERAL - 10
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT_P, TERM]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT_P, TERM_P, FACTOR]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT_P, TERM_P, identifier]
[MATCH] - IDENTIFIER - i
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), OP_STMT_P]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), ++]
[MATCH] - INCREMENT OPERATOR - ++
[MATCH] - RIGHT PARENTHESIS - )
[GENERATE-stack] [$, end, STMTS_P, end, STMTS, begin]
[MATCH] - KEYWORD - begin
```

```
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, PRINT_STMT]
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), PRINT_CONTENT, (, display]
[MATCH] - KEYWORD - display
[MATCH] - LEFT PARENTHESIS - (
[GENERATE-stack] [$, end, STMTS_P, end, STMTS_P, ;, ), string_literal]
[MATCH] - STRING_LITERAL - "test input"
[MATCH] - RIGHT PARENTHESIS - )
[MATCH] - STATEMENT TERMINATOR - ;
[GENERATE-stack] [$, end, STMTS_P, end]
[MATCH] - KEYWORD - end
[GENERATE-stack] [$, end, STMTS]
[GENERATE-stack] [$, end, STMTS_P, STMT]
[GENERATE-stack] [$, end, STMTS_P, WHILE_STMT]
[GENERATE-stack] [$, end, STMTS_P, BEGIN_STMT, ), COM_STMT, (, while]
[MATCH] - KEYWORD - while
[MATCH] - LEFT PARENTHESIS - (
ERROR: temp not declared
Parsing failed.
```