**Project 1: Scanner**

2016024875 손정우

**1. Compilation methods and environment**

To compile the project, I used Makefile example given by the TA. The environment that I build this project was as below:

* Ubuntu 16.04.7 (64-bits, on VMware Workstation)
* gcc 5.4.0 20160609
* flex 2.6.0

**2. Implementation of C-scanner**

**1) Using C-code**

In this implementation, I focused on modifying globals.h, main.c, util.c, and scan.c as project specification suggested. (Note that Tiny’s reserved words were left unchanged as specification hints did. I will remove these words in future projects.) Among these source codes, scan.c was the most important as it has getToken() function.

Processing the comment section was the trickiest part since we need to check whether it is just ‘/’ or ‘/\*’. Also, note that any characters can exist between the ‘/\*’ and ‘\*/’, so we need to consider the transition between the INCOMMENT\_ state and INCOMMENT state. The result of solving these problems is as the following code.

/\* scan.c \*/

TokenType getToken(void)

{  /\* 중략 \*/

   while (state != DONE)

   { int c = getNextChar();

     save = TRUE;

     switch (state)

     { case START:

         if (isdigit(c))

           state = INNUM;

         else if (isalpha(c))

           state = INID;

         else if ((c == ' ') || (c == '\t') || (c == '\n'))

           save = FALSE;

         else if (c == '/')

         { save = FALSE;

           state = INOVER;

         }

         else if (c == '=')

           state = INEQ;

         else if (c == '!')

           state = INNE;

         else if (c == '>')

           state = INLT;

         else if (c == '<')

           state = INGT;

         else

         { state = DONE;

           switch (c)

           {

/\* 중략 \*/

           }

         }

         break;

       case INOVER:

         save = FALSE;

         if (c == '\*')

           state = INCOMMENT\_;

         else

         { save = TRUE;

           state = DONE;

           ungetNextChar();

           currentToken = OVER;

         }

         break;

       case INCOMMENT\_:

         save = FALSE;

         if (c == '\*')

           state = INCOMMENT;

         else

           state = INCOMMENT\_;

         break;

       case INCOMMENT:

         save = FALSE;

         if (c == EOF)

         { state = DONE;

           currentToken = ENDFILE;

         }

         else if (c == '/')

           state = START;

         else if (c == '\*')

           state = INCOMMENT;

         else

           state = INCOMMENT\_;

         break;

       case INEQ:

         save = FALSE;

         state = DONE;

         if (c == '=')

           currentToken = EQ;

         else

         { /\* backup in the input \*/

           ungetNextChar();

           save = TRUE;

           currentToken = ASSIGN;

         }

         break;

       /\* 중략 \*/

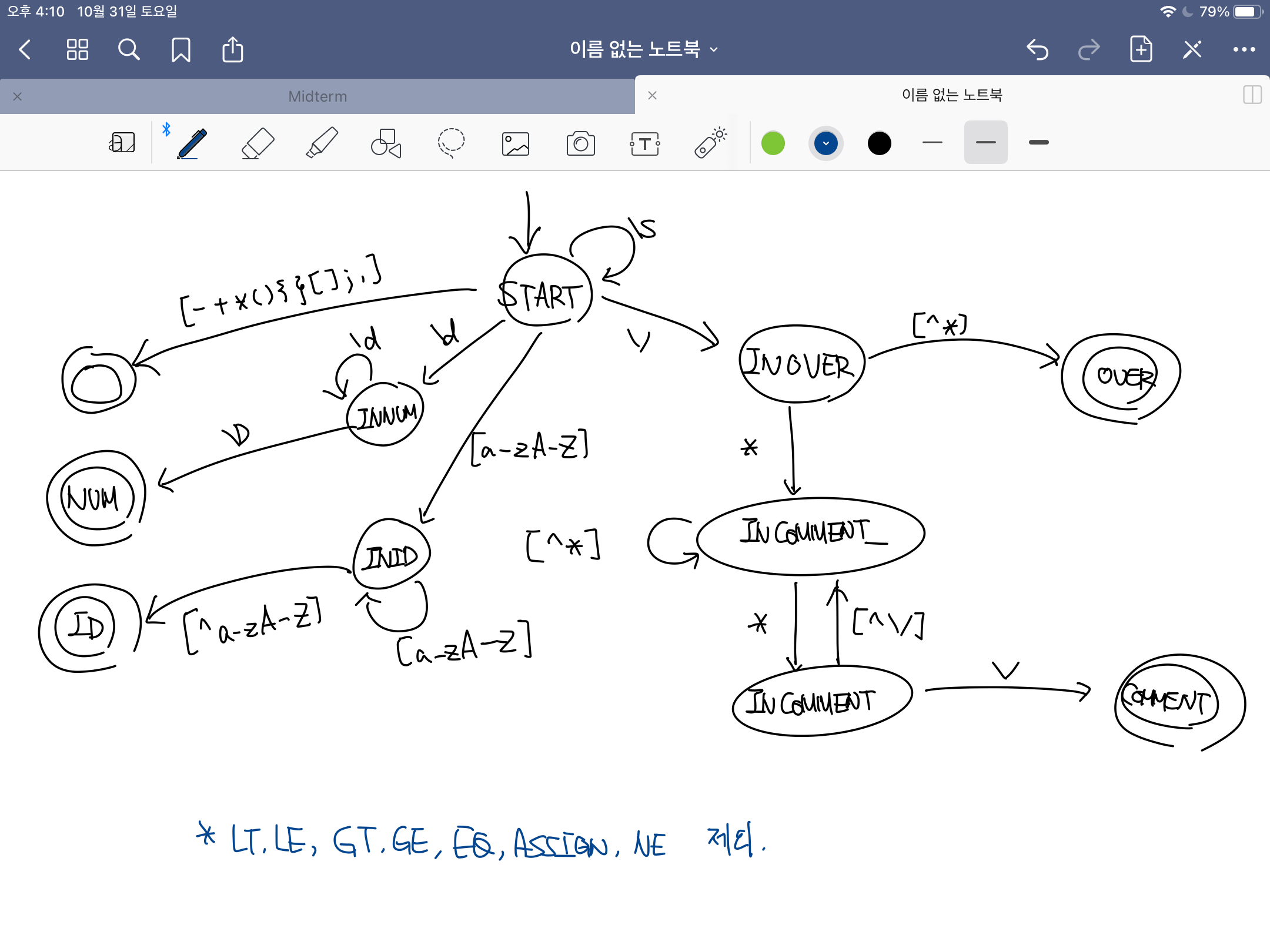
     }

     /\* 중략 \*/

} /\* end getToken \*/

**2) Using flex by Tiny.l**

To change Tiny.l to cminus.l, I first approached by drawing an automata that processes cminus like below.



Based on this automata, I implemented cminus.l like below.

/\* cminus.l \*/

/\* 중략 \*/

%%

"if" {return IF;}

"else" {return ELSE;}

"while" {return WHILE;}

"return" {return RETURN;}

"int" {return INT;}

"void" {return VOID;}

"then" {return THEN;} /\* discarded \*/

"end" {return END;} /\* discarded \*/

"repeat" {return REPEAT;} /\* discarded \*/

"until" {return UNTIL;} /\* discarded \*/

"read" {return READ;} /\* discarded \*/

"write" {return WRITE;} /\* discarded \*/

"=" {return ASSIGN;}

"==" {return EQ;}

"!=" {return NE;}

"<" {return LT;}

"<=" {return LE;}

">" {return GT;}

">=" {return GE;}

"+" {return PLUS;}

"-" {return MINUS;}

"\*" {return TIMES;}

"/" {return OVER;}

"(" {return LPAREN;}

")" {return RPAREN;}

"[" {return LBRACE;}

"]" {return RBRACE;}

"{" {return LCURLY;}

"}" {return RCURLY;}

";" {return SEMI;}

"," {return COMMA;}

{number} {return NUM;}

{identifier} {return ID;}

{newline} {lineno++;}

{whitespace} {/\* skip whitespace \*/}

"/\*" { char b, c;

b = 0;

do

{ c = input();

if (c == EOF) break;

else if (c == '\n') lineno++;

else if (b == '\*' && c == '/') break;

b = c;

} while (TRUE);

}

. {return ERROR;}

%%

/\* 후략 \*/

**3. Example Results**

By referring to the provided example file and the result value, I confirmed that the codes I implemented work normally. The following are screen shots showing that the code works ordinarily.

|  |  |  |
| --- | --- | --- |
| **Testcase** | **Using C codes** | **Using flex** |
| test.1.txt |  |  |
| test.2.txt |  |  |
| \*Note that some of the results have been omitted to meet the report specifications. (less than 5 pages) | | |