EXP NO: 5 **DATE:**

RECOGNIZE A VALID VARIABLE WHICH STARTS WITH A LETTER FOLLOWED BY ANY NUMBER OF LETTERS OR DIGITS USING LEX AND YACC

Problem Statement:
Recognizes a valid variable name. The variable name must start with a letter (either uppercase or lowercase) and can be followed by any number of letters or digits. The program should validate whether a given string adheres to this naming convention.

AIM:
To develop a **LEX and YACC program** that recognizes a **valid variable name** in C programming, which:
 Starts with a **letter** (a-z or A-Z)
 Followed by **any number of letters or digits** (a-z, A-Z, 0-9, _)
 Does not allow invalid characters (e.g., 123abc, @var, x!y)

220701215

ALGORITHM:

1. A Yacc source program has three parts as follows: Declarations %% translation rules %% supporting C routines
2. Declarations Section: This section contains entries that:
    - Include standard I/O header file.
    - Define global variables.
    - Define the list rule as the place to start processing.
    - Define the tokens used by the parser.
3. Rules Section: The rules section defines the rules that parse the input stream. Each rule of a grammar production and the associated semantic action.
4. Programs Section: The programs section contains the following subroutines. Because these subroutines are included in this file, it is not necessary to use the yacc library when processing this file.
    - Main- The required main program that calls the yyparse subroutine to start the program.
    - yyerror(s) -This error-handling subroutine only prints a syntax error message.
    - yywrap -The wrap-up subroutine that returns a value of 1 when the end of input occurs. The
    - calc.lex file contains include statements for standard input and output, as programmer file
    - information if we use the -d flag with the yacc command. The y.tab.h file contains definitions
    - for
    - the tokens that the parser program uses.
5. calc.lex contains the rules to generate these tokens from the input stream.

PROGRAM:

LEX PROGRAM

```
%{
#include "y.tab.h"
%}
%option noyywrap
%%
// Pattern for valid variable names
[a-zA-Z][a-zA-Z0-9]* { return IDENTIFIER; }
// Ignore whitespace
[ \t\n] { /* Skip */ }
. { return yytext[0]; }
%%
```

YACC PROGRAM

```
%{
#include <stdio.h>
void yyerror(const char *msg);
%}
%token IDENTIFIER
%%
stmt: IDENTIFIER { printf("Valid variable: %s\n", yytext); }
;
%%
void yyerror(const char *msg) {
printf("Invalid variable\n");
}
35
int main() {
printf("Enter a variable name: ");
yyparse();
return 0;
}
```

220701215

OUTPUT :

```
yacc -d parser.y
lex lexer.l
cc lex.yy.c y.tab.c -o var_checker
./a.out
```

```
$ ./var_checker
Enter a variable name: myVar123
Valid variable: myVar123
```

RESULT: Thus the above program reads an input string, checks whether it follows the rules for a valid variable name, and produces the following output.