EXP NO-5

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

AIM:

To design and implement a Desk Calculator using the LEX tool, which validates arithmetic expressions containing +, -, *, /, numbers, and parentheses. The program ensures that the expression follows correct arithmetic syntax rules.

PROGRAM: VAR.L

```
%{
#include "y.tab.h"
%}

%option noyywrap

%%

[a-zA-Z_][a-zA-Z0-9_]* { return IDENTIFIER; }
[ \t\n] { /* Skip */ }
. { return yytext[0]; }

%%
```

VAR.Y

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

OUTPUT

```
kamali@Kamali:~$ yacc -d var.y
kamali@Kamali:~$ lex var.l
kamali@Kamali:~$ gcc lex.yy.c y.tab.c -o par
y.tab.c: In function ‘yyparse’:
y.tab.c:1010:16: warning: implicit declaration of function ‘yylex’ [-Wimplicit-function-declaration]
 1010 |       yychar = yylex ();
      |                ^~~~~
kamali@Kamali:~$ ./par
Enter a variable name: myVar1
Valid variable: myVar1
1Sb
Invalid character
kamali@Kamali:~$
```

RESULT:

Thus the above program reads an arithmetic expression, tokenizes it using LEX rules, and validates the syntax by recognizing numbers, operators (+, -, *, /), and parentheses. If the expression is valid, it prints "Valid arithmetic expression." Otherwise, it detects and reports invalid tokens.

KAMALI K A -220701118