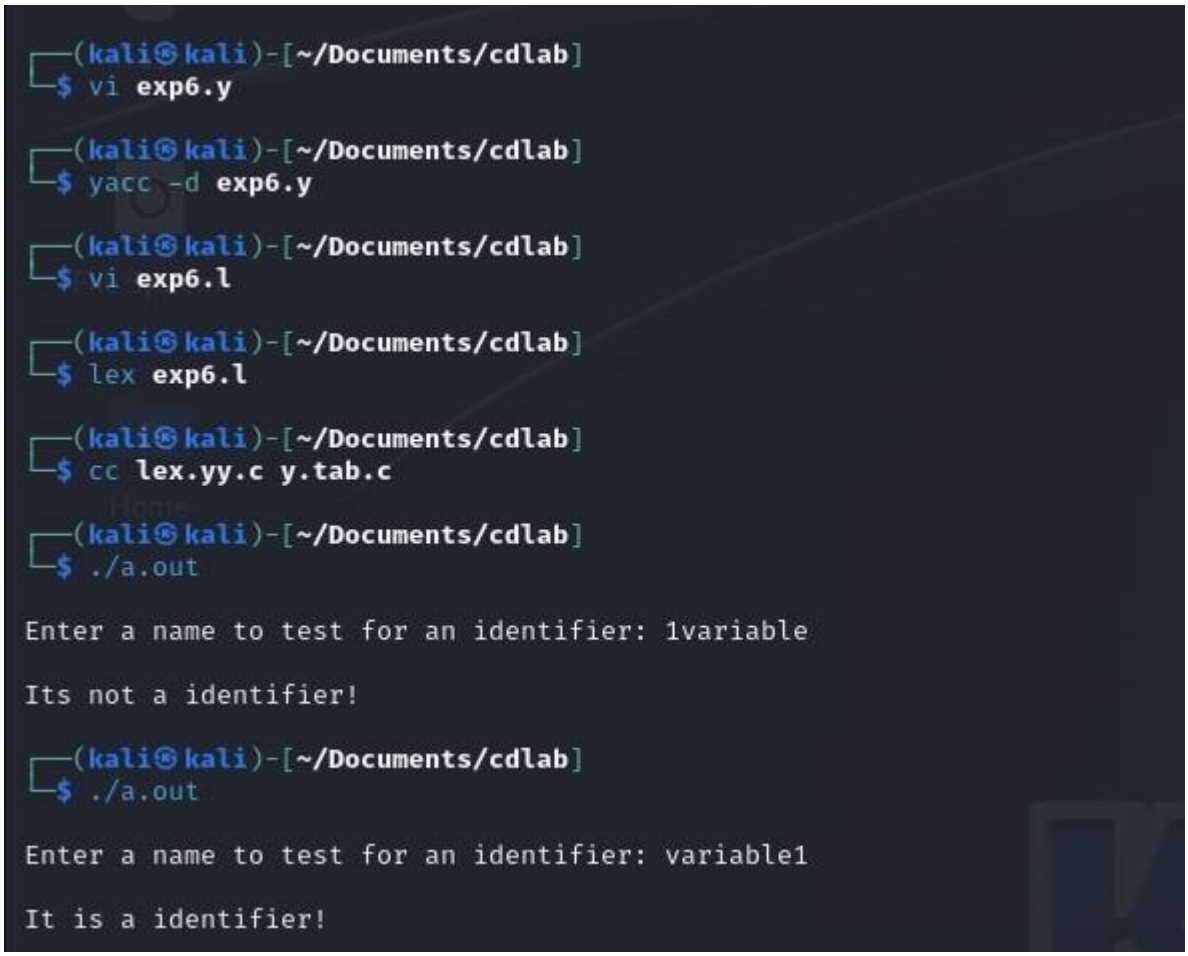
OUTPUT:



RESULT:

Thus, a program using lex and yacc tool is implemented to recognize a valid variable which starts with a letter followed by any number of letters or digits.

Roll Number: 210701075
Name: Harish R

Ex No: 7
Date:

EVAUATE EXPRESSION THAT TAKES DIGITS, *, + USING LEX AND YACC

AIM:

To perform arithmetic operations that takes digits,*, + using lex and yacc.

ALGORITHM:

Lex (exp7.l):

1. Recognizes sequences of digits and returns the token NUMBER.
2. Ignores tabs and newlines.
3. Returns any other single character as itself.
4. Indicates the end of input with yywrap().

Yacc (exp7.y):

1. Includes headers and declares global variables.
2. Declares token NUMBER.
3. Defines operator precedence and associativity.
4. Defines grammar rules for arithmetic expressions.
5. Prints the result of the expression evaluation in the ArithmeticExpression rule.
6. Handles syntax errors with yyerror().
7. The main function, prompts for an arithmetic expression, parses it, and prints whether it's valid or not based on the presence of syntax errors.

PROGRAM:

exp7.l:

```
%{
#include<stdio.h>
#include "y.tab.h" extern
int yylval;
%}
%%
[0-9]+ {
```

```
        yylval=atoi(yytext);
        return NUMBER;
        }
[\t] ;
[\n] return 0;
. return yytext[0];
%%
int yywrap()
{
return 1;
}
```

exp7.y:

```
%{
        #include<stdio.h>
        int flag=0;
  int yylex();  void yyerror();
%}
%token NUMBER
%left '+' '-'
%left '*' '/' '%'
%left '(' ')'
%%
ArithmeticExpression: E{
        printf("\nResult=%d\n",$$);
        return 0;
        }
E:E'+'E {$$=$1+$3;}   |E'-
'E {$$=$1-$3;}
 |E'*'E {$$=$1*$3;}
```

Roll Number: 210701075
Name: Harish R