

Exp No: 5

Date:

DESIGN A DESK CALCULATOR USING LEX TOOL

AIM:

To check whether the arithmetic expression using lex and yacc tool.

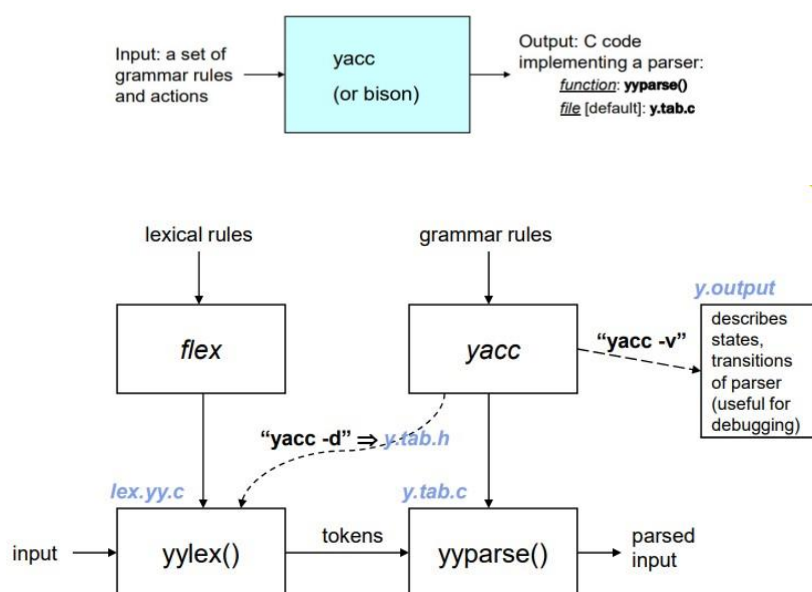
ALGORITHM:

- Using the flex tool, create lex and yacc files.
- In the C include section define the header files required.
- In the rules section define the REGEX expressions along with proper definitions.
- In the user defined section define yywrap() function.
- Declare the yacc file inside it in the C definitions section declare the header files required along with an integer variable valid with value assigned as 1.
- In the Yacc declarations declare the format token num id op.
- In the grammar rules section if the starting string is followed by assigning operator or identifier or number or operator followed by a number or open parenthesis followed by an identifier. The x could be an operator followed by an identifier or operator or no operator then declare that as valid expressions by making the valid stay in 1 itself.
- In the user definition section if the valid is 0 print as Invalid expression in yyerror() and define the main function.

LEX AND YACC WORKING:

Parser generator:

- Takes a specification for a context-free grammar.
- Produces code for a parser.



Name: Keerthiga K

Roll No: 210701120

PROGRAM:

cdlab5.l:

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

%%

[a-zA-Z_][a-zA-Z_0-9]* return id;

[0-9]+(\.[0-9]*)?    return num;

[+/*]                return op;

.                    return yytext[0];
\n                    return 0;
%%
```

```
int yywrap(){
return 1; }
```

cdlab5.y:

```
%{
    #include<stdio.h>
    int yylex();    int
    yyerror();
    int valid=1;
```

```
%}
```

```
%token num id op
```

```
%%
```

```
start : id '=' s
      ; s :    id x
          | num x
          | '-' num x
      | '(' s ')' x
      ; x :    op s
          | '-' s
          |
```

Name: Keerthiga K

Roll No: 210701120

```

;
%%
int yyerror(){ valid=0;
printf("\nInvalid expression!\n");
return 0;
} int
main(){
printf("\nEnter the expression:\n");
yyparse(); if(valid){
printf("\nValid expression!\n");
}}

```

OUTPUT:

```

(kali㉿kali)-[~/Documents/cdlab]
$ vi cdlab5.y

(kali㉿kali)-[~/Documents/cdlab]
$ yacc -d cdlab5.y

(kali㉿kali)-[~/Documents/cdlab]
$ vi cdlab5.l

(kali㉿kali)-[~/Documents/cdlab]
$ lex cdlab5.l

(kali㉿kali)-[~/Documents/cdlab]
$ gcc lex.yy.c y.tab.c

(kali㉿kali)-[~/Documents/cdlab]
$ ./a.out

Enter the expression:
a=b

Invalid expression!

(kali㉿kali)-[~/Documents/cdlab]
$ ./a.out

Enter the expression:
a=b;

Valid expression!

```

RESULT:

Thus, a program to check whether the arithmetic expression using lex and yacc tool is implemented.