Expt. No. 4 Page No. 11

Expt. Name. Use LEX & YACC tool to implement Desktop Calculator Date :

Aim:

To write the program using LEX and YACC to implement ~~parser or~~ Desktop Calculator

Algorithm:

File.l

step 1: start
step 2: Include the necessary header files and declare the necessary variables
step 3: Define the keywords and the identifiers with the constant and operator
step 4: Get the input for analysis from user
step 5: check each and every element in the statement with number
step 6: check each and every element in the statement with small alphabet
step 7: check each and every element for the operator
step 8: Else print Invalid Token
step 9: return the value
step 10: stop

File.y

step 1: start
step 2: Include the necessary header files and declare the necessary variables
step 3: Define the keywords and the identifiers with the constant and operator.
step 4: Take the value which was taken from user and implement the respective operator.
step 5: return the value and print it.

Step 6: Stop

Program: → File.l

```
%{
    #include <stdlib.h>
    #include "y.tab.h"
    void yyerror(char *s);
    extern int yylval;
%}
%%
[0-9]+      {yylval = atoi(yytext); return INT;}
[a-z]       {yylval = toascii(*yytext)-97;} return ID;}
[A-Z]       {yylval = to ascii(*yytext)-65; return ID;}
[-+*=\n]    {return *yytext;}
'('         {return *yytext;}
')'         {return *yytext;}
[\t] ;
.           {yyerror("Invalid Token!!");}
%%
int yywrap()
{
return 1;
}
```

File.y

```
%{
#include <stdio.h>
extern int yylex(void);
void yyerror(char *);
```

Expt. No. _____ Page No. 13

Expt. Name. _____ Date : _____

```
int x=0
int val[26];
%}
% token INT ID
%%
Mohnish:
 mohnish expr '\n'          { x=$2; printf("%d\n", $2);}
|mohnish ID '=' expr '\n'    { val[$2]=$4;}
|
;
 expr:
 expr '+' T                 {$$ = $1+$3;}
|expr '-' T                 {$$= $1-$3;}
| T                         {$$ = $1;}
| '+' T                     {$$ = x+$2;}
| '-' T                     {$$ = x-$2;}
;
T:
 F                          {$ $= $1;}
| T '*' F                   {$$ = $1*$3;}
| T '/' F                   {$$ = $1 / $3;}
| '*' F                     {$$ = x*$2;}
| '/' F                     {$$ = x/$2;}
;
 INT                        {$$ = $1;}
|ID                         {$$ = val[$1];}
| '(' expr ')'              {$$ = $2;}
 ;
%%
```

```
void yyerror (char* s)
{
printf ("%s", s);
}

int main()
{
yyparse();
return 0;
}
```

Result:

Use LEX and YACC to implement ~~parsere for~~ Desktop Calculator is executed successfully.