**Experiment 8**

**Aim:** Write a program to make simple calculator using Bison.

**Input of program-** Any Valid/Invalid Arithmetic Operations.

**Output of program-** Result of Input.

**Tools:** gcc compiler, Text editor, flex, bison, make.

**Procedure:**

**Flex Code:**

%{

#include <stdlib.h>

void yyerror(char \*);

#include "y.tab.h"

extern int yylval;

%}

%%

[0-9]+ {

yylval = atoi(yytext);

return INTEGER;

}

"^" return \*yytext;

[-+\*/\n] return \*yytext;

[ \t] ; /\* skip whitespace \*/

. yyerror("invalid character");

%%

int main(void)

{

yyparse();

return 0;

}

int yywrap(void) {

return 1;

}

void yyerror(char \*s) {

fprintf(stderr, "%s\n", s);

}

**Bison Code:**

**Cal.y**

%{

#include <stdio.h>

#include <math.h>

int yylex(void);

int getans(double x,double y);

void yyerror(char \*);

%}

%token INTEGER

%left '-' '+'

%left '\*' '/'

%right '^'

%%

program :

program expr '\n' { printf("%d\n", $2); }

|

;

expr : expr '+' term { $$ = $1 + $3; }

| expr '-' term { $$ = $1 - $3; }

| term { $$ = $1; }

;

term : term '\*' factor { $$ = $1 \* $3; }

| term '/' factor { $$ = $1 / $3; }

| factor { $$ = $1; }

;

factor : factor '^' num {$$ = pow($1,$3);}

| num {$$ = $1;}

;

num : '(' expr ')' { $$ = $2; }

| INTEGER

;

//%%

Makefile Code:

Run:

flex cal.l

bison -dy cal.y

gcc lex.yy.c y.tab.c -o cal -lm

./cal

**Output:-**

