**­**

|  |  |
| --- | --- |
| **Total Marks:** | **04** |
| **Obtained Marks:** |  |

**Compiler Construction**

**Assignment # 04**

**Last date of Submission: 13 May 2025**

**Submitted To: Safi Ullah**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Student Name: Ubaid Bin Waris**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Reg Number: 2212416**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

***Instructions****: Copied or shown assignments will be marked zero. Late submissions are not entertained in any case.*

Develop a command-line scientific calculator using Lex and Yacc that supports:

1. Basic Operations:

- Arithmetic: +, -, \*, /, mod

- Exponentiation: ^ or pow()

2. Scientific Functions:

- Trigonometry: sin, cos, tan, asin, acos, atan, cot, sec, csc

- Logarithms: log (natural), log10, ln

- Other: exp, sqrt

3. Constants: pi (π), e (Euler’s number)

4. Variables:

- Single-letter, case-sensitive (e.g., A and a are distinct).

- Support assignment (e.g., x = 5) and retrieval.

5. Output: A print command to display results/variables.

6. Error Handling: Division/modulo by zero, Invalid expressions and Undefined variables.

7. Test Data:

(3+4)\*sin(pi/3);

A=(2+3)\*4;

A=(3-2)\*4;

Print a;

Print A;

**Note:**

1. Rename the file to your ID (e.g., 2073105.docx), Upload it on Google Classroom and submit a single-sided hard copy in class.
2. Include the source code, compilation screen, and runtime screen in the document.
3. Ensure the output screen has a plain background (no black or colored background).
4. Maintain proper indentation and formatting to avoid mark deductions.
5. Group work is not allowed. Each student must complete and submit their assignment individually.

**Solution**

**calc.l**

%{

#include "y.tab.h"

#include <math.h>

#include <stdlib.h>

#include <string.h>

%}

digit [0-9]

letter [a-zA-Z]

%%

"pi" { yylval.fval = M\_PI; return NUMBER; }

"e" { yylval.fval = M\_E; return NUMBER; }

"sin" { return SIN; }

"cos" { return COS; }

"tan" { return TAN; }

"asin" { return ASIN; }

"acos" { return ACOS; }

"atan" { return ATAN; }

"cot" { return COT; }

"sec" { return SEC; }

"csc" { return CSC; }

"log10" { return LOG10; }

"log"|"ln" { return LOG; }

"sqrt" { return SQRT; }

"exp" { return EXP; }

"pow" { return POW; }

"mod" { return MOD; }

[Pp][Rr][Ii][Nn][Tt] { return PRINT; }

{letter} { yylval.sval = strdup(yytext); return VAR; }

{digit}+("."{digit}+)? { yylval.fval = atof(yytext); return NUMBER; }

"=" { return ASSIGN; }

";" { return SEMICOLON; }

"^" { return POW; }

[()+\-\*/] { return \*yytext; }

[ \t\n] ; // ignore whitespace

. { printf("Invalid character: %s\n", yytext); return INVALID; }

%%

int yywrap(void) { return 1; }

**calc.y**

%{

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#define MAXVARS 52

typedef struct {

char name;

double value;

int defined;

} Variable;

Variable variables[MAXVARS];

double get\_var(char \*name) {

char ch = name[0];

int idx = (ch >= 'A' && ch <= 'Z') ? ch - 'A' : ch - 'a' + 26;

if (!variables[idx].defined) {

printf("Error: Variable '%c' is undefined.\n", ch);

return 0;

}

return variables[idx].value;

}

void set\_var(char \*name, double value) {

char ch = name[0];

int idx = (ch >= 'A' && ch <= 'Z') ? ch - 'A' : ch - 'a' + 26;

variables[idx].name = ch;

variables[idx].value = value;

variables[idx].defined = 1;

}

void print\_var(char \*name) {

char ch = name[0];

int idx = (ch >= 'A' && ch <= 'Z') ? ch - 'A' : ch - 'a' + 26;

if (variables[idx].defined)

printf("%c = %.6f\n", ch, variables[idx].value);

else

printf("Error: Variable '%c' is undefined.\n", ch);

}

int yylex(void);

void yyerror(const char \*s) {

printf("Error: %s\n", s);

}

%}

%union {

double fval;

char \*sval;

}

%token <fval> NUMBER

%token <sval> VAR

%token PRINT ASSIGN SEMICOLON INVALID

%token SIN COS TAN ASIN ACOS ATAN COT SEC CSC

%token LOG LOG10 SQRT EXP POW

%token MOD

%left '+' '-'

%left '\*' '/' MOD

%right '^' POW

%right UMINUS

%type <fval> expr

%%

program:

program statement

|

;

statement:

expr SEMICOLON { printf("= %.6f\n", $1); }

| VAR ASSIGN expr SEMICOLON { set\_var($1, $3); free($1); }

| PRINT VAR SEMICOLON { print\_var($2); free($2); }

;

expr:

NUMBER { $$ = $1; }

| VAR { $$ = get\_var($1); free($1); }

| expr '+' expr { $$ = $1 + $3; }

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

| expr '\*' expr { $$ = $1 \* $3; }

| expr '/' expr {

if ($3 == 0) {

yyerror("Division by zero");

$$ = 0;

} else {

$$ = $1 / $3;

}

}

| expr MOD expr {

if ((int)$3 == 0) {

yyerror("Modulo by zero");

$$ = 0;

} else {

$$ = (int)$1 % (int)$3;

}

}

| expr '^' expr { $$ = pow($1, $3); }

| POW '(' expr ',' expr ')' { $$ = pow($3, $5); }

| SIN '(' expr ')' { $$ = sin($3); }

| COS '(' expr ')' { $$ = cos($3); }

| TAN '(' expr ')' { $$ = tan($3); }

| ASIN '(' expr ')' { $$ = asin($3); }

| ACOS '(' expr ')' { $$ = acos($3); }

| ATAN '(' expr ')' { $$ = atan($3); }

| COT '(' expr ')' { $$ = 1.0 / tan($3); }

| SEC '(' expr ')' { $$ = 1.0 / cos($3); }

| CSC '(' expr ')' { $$ = 1.0 / sin($3); }

| LOG '(' expr ')' { $$ = log($3); }

| LOG10 '(' expr ')' { $$ = log10($3); }

| SQRT '(' expr ')' { $$ = sqrt($3); }

| EXP '(' expr ')' { $$ = exp($3); }

| '-' expr %prec UMINUS { $$ = -$2; }

| '(' expr ')' { $$ = $2; }

;

%%

int main(void) {

printf("Scientific Calculator (type expressions and end with ;)\n");

printf("Enter 'Print x;' to view variable x.\n\n");

return yyparse();

}

**Output**

A computer screen with text

AI-generated content may be incorrect.