CD FISAC

Section: CSE C

| Name | Roll Number | Registration Number |
| --- | --- | --- |
| Rithika Bhat | 1 | 210905002 |
| Poorvi J Nayak | 3 | 210905006 |
| Aditi Bhoomireddy | 4 | 210905007 |
| Kshiti Shetty | 25 | 210905137 |
| Himanshu Banerji | 32 | 210905180 |

flex.l :

%{

#include "bison.tab.h"

%}

digit [0-9]

letter [a-zA-Z]

%%

"," { return COMMA; }

"union" { return UNION; }

"struct" { return STRUCT; }

"static" { return STATIC; }

"const" { return CONST; }

"int" { return INT; }

"float" { return FLOAT; }

"char" { return CHAR; }

"double" { return DOUBLE; }

"bool" { return BOOL; }

"return" { return RETURN; }

"if" { return IF; }

"else" { return ELSE; }

"while" { return WHILE; }

[a-zA-Z\_][a-zA-Z0-9\_]\* { return IDENTIFIER; }

[0-9][0-9]\*[.][0-9][0-9]\* { return FLOAT\_CONSTANT; }

[0-9][0-9]\* { return INTEGER\_CONSTANT; }

"==" { return EQ; }

"!=" { return NEQ; }

"<" { return LT; }

">" { return GT; }

"<=" { return LTEQ; }

">=" { return GTEQ; }

"+" { return PLUS; }

"-" { return MINUS; }

"\*" { return MULT; }

"/" { return DIV; }

"=" { return ASSIGN; }

";" { return SEMICOLON; }

"(" { return LPAREN; }

")" { return RPAREN; }

"{" { return LBRACE; }

"}" { return RBRACE; }

"\"" { return STRING\_CONSTANT; }

'[^"\\]+' { return STRING\_CHARACTER; }

"\\" { return ESCAPE\_SEQUENCE; }

. { /\* ignore other characters \*/ }

%%

int yywrap() {}

bison.y:

%{

#include <stdio.h>

#include<stdlib.h>

extern FILE\* yyin;

%}

%token COMMA IDENTIFIER UNION STRUCT STATIC CONST INT FLOAT CHAR DOUBLE BOOL RETURN IF ELSE WHILE EQ NEQ LT GT LTEQ GTEQ PLUS MINUS MULT DIV ASSIGN SEMICOLON LPAREN RPAREN LBRACE RBRACE STRING\_CONSTANT STRING\_CHARACTER

%token INTEGER\_CONSTANT FLOAT\_CONSTANT ESCAPE\_SEQUENCE

%%

program: struct\_declaration { printf("Successful compilation"); exit(0) };

| union\_declaration { printf("Successful compilation"); exit(0) };

struct\_declaration: STRUCT identifier LBRACE member\_list RBRACE SEMICOLON

| STRUCT identifier SEMICOLON

;

member\_list: member\_declaration

| member\_list member\_declaration

;

member\_declaration: STATIC type identifier ASSIGN expression SEMICOLON

| CONST type identifier ASSIGN expression SEMICOLON

| type identifier ASSIGN expression SEMICOLON

| type identifier SEMICOLON

| function\_declaration

| union\_declaration

| struct\_declaration

;

function\_declaration: type identifier LPAREN parameter\_list RPAREN LBRACE declarations statements RBRACE

;

declarations: declaration

| declarations declaration

| /\*empty\*/

;

declaration: type identifier SEMICOLON

| type identifier ASSIGN expression SEMICOLON

;

parameter\_list: parameter

| parameter\_list COMMA parameter

| /\*empty\*/

;

parameter: type identifier

;

statements: statement

| statements statement

| /\*empty\*/

;

statement: expression SEMICOLON

| if\_statement

| while\_statement

| return\_statement

| block

;

if\_statement: IF LPAREN condition RPAREN statement ELSE statement

;

while\_statement: WHILE LPAREN condition RPAREN statement

;

return\_statement: RETURN expression SEMICOLON

;

block: LBRACE statements RBRACE

;

condition: expression relational\_operator expression

;

relational\_operator: EQ

| NEQ

| LT

| GT

| LTEQ

| GTEQ

;

type: INT

| FLOAT

| CHAR

| DOUBLE

| BOOL

| identifier

;

expression: term

| expression PLUS term

| expression MINUS term

| assignment\_statement

;

assignment\_statement: identifier ASSIGN expression

| type identifier ASSIGN expression

identifier: IDENTIFIER

;

union\_declaration: UNION identifier LBRACE member\_list RBRACE SEMICOLON

;

term: factor

| term MULT factor

| term DIV factor

;

factor: identifier

| LPAREN expression RPAREN

| constant

;

constant: integer\_constant

| float\_constant

| STRING\_CONSTANT

;

integer\_constant: INTEGER\_CONSTANT

| MINUS INTEGER\_CONSTANT

;

float\_constant: FLOAT\_CONSTANT

| MINUS FLOAT\_CONSTANT

;

%%

int yyerror(const char \*msg) {

fprintf(stderr, "Error: %s\n", msg);

exit(0);

}

int main() {

yyin=fopen("input.cpp","r");

yyparse();

return 0;

}

Sample Input:

struct a

{

int e;

int d;

void sum()

{

int c;

c=e+1;

}

union \_abc123

{

float exp;

struct a12

{

int returnDiff()

{

int xyz=10;

int exp=exp-1;

return exp;

}

};

};

};

Sample output:

'Successful Compilation'