# PES2UG20CS237_CD_ASSIGNMENT1

## Details :

- Name : P K Navin Shrinivas
- Section : D
- SRN : PES2UG20CS237

## Lexer and yacc

### Lexer.l :

```language-c
%{
#include<stdio.h>
#include "y.tab.h"
void yyerror(char *s);
int yylineno;
%}
letter [a-zA-Z_]
digit [0-9]
sign [+-]?
fraction (\.{digit}+)?
exp ([Ee][+-]{digit}+)?
number {sign}{digit}*{fraction}{exp}
id {letter}({letter}|{digit})*
%x state
%%
"//".* ;
\/\* {yymore(); BEGIN state;}
<state>' '|\t {yymore(); BEGIN state;}
<state>\n {yymore(); ++yylineno; BEGIN state;}
<state>^\* {yymore(); BEGIN state;}
<state>"*"[^/] {yymore(); BEGIN state;}
<state>"*"\/ BEGIN 0 ;
main return MAIN;
int return INT;
```

```
char return CHAR;
float return FLOAT;
double return DOUBLE;
for return FOR;
do return DO;
while return WHILE;
if return IF;
else return ELSE;
#include return INCLUDE;
{id} return ID;
"+" return *yytext;
"-" return *yytext;
{number} return NUMBER;
{id}\.h return HEADER;
"++" return INC;
"--" return DEC;
"≥" return GREATEREQ;
"≤" return LESSEREQ;
"==" return EQCOMP;
"≠" return NOTEQ;
"&&" return ANDAND;
"||" return OROR;
\r ;
\t ;
[ ] ;
\n { ++yylineno; };
. return *yytext;
%%
int yywrap()
{
return(1);
}
```

**parser.y :**

```
%{                                                    language-c
#include<stdio.h>
#include<stdlib.h>
int yylex();
void yyerror(char *s);
extern int yylineno;
extern char *yytext;
```

```
%{
%token INT FLOAT DOUBLE CHAR FOR WHILE DO IF ELSE INCLUDE MAIN ID NUMBER
HEADER
GREATEREQ LESSEREQ EQCOMP NOTEQ INC DEC ANDAND OROR
%left '+' '-'
%left '*' '/'
%%
Start : Prog { printf("Declarations are valid.\n"); YYACCEPT; };
Prog: INCLUDE '<' HEADER '>' Prog | MainF Prog | Declr ';' Prog | Assgn
';' Prog |
ArrayDecl ';' Prog | error ';' {yyerrok;yyclearin;} Prog |;
ArrayDecl: ID Bracket;
Bracket: '[' NUMBER ']' Bracket| '[' ID ']'Bracket| ;
Declr: Type ListVar;
ListVar: ListVar ',' ID | InitDeclr | ArrayDecl | ID;
InitDeclr: Assgn ',' InitDeclr | Assgn;
Type: INT | FLOAT | DOUBLE | CHAR;
Unary_operator: '&' | '*' | '+' | '-' | '~' | '!';
IncDec: INC | DEC ;
Assgn: ID '=' Expr | ID '=' Logical | ArrayDecl '=' Expr | ArrayDecl '='
Logical;
Logical: ID ANDAND Logical | ID OROR Logical | ID;
Expr: Expr Relop E | Unary_operator ID | ID IncDec | E;
Relop: '<' | '>' | LESSEREQ | GREATEREQ | EQCOMP | NOTEQ;
E: E '+' T | E '-' T | T;
T: T '*' F | T '/' F | F;
F: '(' Expr ')' | ID | NUMBER;
MainF: Type MAIN '(' Empty_ListVar ')' '{' Stmt '}';
Empty_ListVar: ListVar | ;
Stmt: SingleStmt Stmt | Block Stmt | ;
SingleStmt: Declr ';' | Assgn ';' | Cond ';' | IF '(' Cond ')' Stmt | IF
'(' Cond
')' Stmt ELSE Stmt | WhileL | ForL | DoWhileL | error ';'
{yyerrok;yyclearin;};
Block: '{' Stmt '}';
WhileL: WHILE '(' Cond ')' Loop_body;
Cond: Expr | Assgn | Logical;
Loop_body: '{' Stmt '}' | ;
multi_expression: Cond | Type Cond | multi_expression ',' Cond;
expression_statement : ';' | multi_expression ';';
ForL: FOR '(' expression_statement expression_statement multi_expression
')'
```

```
  Loop_body;
  DoWhileL: DO Loop_body WHILE '(' Cond ')' ';';
  %%
  void yyerror(char *s)
  {
  printf("Error: %s, Line number: %d, Token: %s\n", s, yylineno, yytext);
  }
  int main()
  {
  if(!yyparse())
  {
  printf("Parsing Successful\n");
  }
  else
  {
  printf("Unsuccessful\n");
  }
  return 0;
  }
```

# Screenshot of valid and invalid cases :

```
➜  lab1 git:(main) ✗ lex PES2UG20CS237.l
➜  lab1 git:(main) ✗ yacc -d PES2UG20CS237.y
PES2UG20CS237.y: warning: 219 shift/reduce conflicts [-Wconflicts-sr]
PES2UG20CS237.y: note: rerun with option '-Wcounterexamples' to generate conflict counterexamples
➜  lab1 git:(main) ✗ ls
invalid_test_lab1.c  PES2UG20CS237_D_LAB1_CD.md  PES2UG20CS237.y    y.tab.c
lex.yy.c             PES2UG20CS237.l             valid_test_lab1.c  y.tab.h
➜  lab1 git:(main) ✗ gcc y.tab.c lex.yy.c
➜  lab1 git:(main) ✗ ls
a.out                lex.yy.c                    PES2UG20CS237.l  valid_test_lab1.c  y.tab.h
invalid_test_lab1.c  PES2UG20CS237_D_LAB1_CD.md  PES2UG20CS237.y  y.tab.c
➜  lab1 git:(main) ✗ ./a.out < valid_test_lab1.c

Valid syntax
➜  lab1 git:(main) ✗ ./a.out < invalid_test_lab1.c
Error: syntax error, line number: 6, token: 1
Error: syntax error, line number: 7, token:
➜  lab1 git:(main) ✗
```