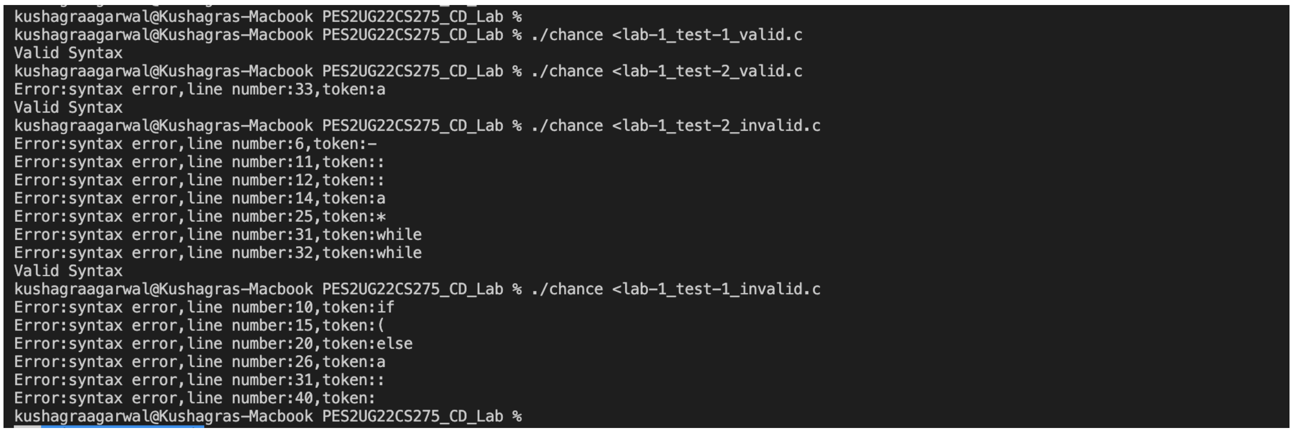
SRN:PES2UG22CS275
NAME:Kushagra Agarwal



```
kushagraagarwal@Kushagras-Macbook PES2UG22CS275_CD_Lab %
kushagraagarwal@Kushagras-Macbook PES2UG22CS275_CD_Lab % ./chance <lab-1_test-1_valid.c
Valid Syntax
kushagraagarwal@Kushagras-Macbook PES2UG22CS275_CD_Lab % ./chance <lab-1_test-2_valid.c
Error:syntax error,line number:33,token:a
Valid Syntax
kushagraagarwal@Kushagras-Macbook PES2UG22CS275_CD_Lab % ./chance <lab-1_test-2_invalid.c
Error:syntax error,line number:6,token:-
Error:syntax error,line number:11,token::
Error:syntax error,line number:12,token::
Error:syntax error,line number:14,token:a
Error:syntax error,line number:25,token:*
Error:syntax error,line number:31,token:while
Error:syntax error,line number:32,token:while
Valid Syntax
kushagraagarwal@Kushagras-Macbook PES2UG22CS275_CD_Lab % ./chance <lab-1_test-1_invalid.c
Error:syntax error,line number:10,token:if
Error:syntax error,line number:15,token:(
Error:syntax error,line number:20,token:else
Error:syntax error,line number:26,token:a
Error:syntax error,line number:31,token::
Error:syntax error,line number:40,token:
kushagraagarwal@Kushagras-Macbook PES2UG22CS275_CD_Lab %
```

Lex file:

```
%{
#include<stdio.h>
#include "y.tab.h"
int yywrap();
int yylineno;
%}


letter [a-zA-Z]
digit [0-9]
id (_|{letter})(_|{letter}|{digit})*
strlit \".*\"
opsign [+-]?
opfrac (\.{digit})?
opexponent ([Ee][+-]?{digit}+)?
number {opsign}{digit}+{opfrac}{opexponent}
start   \/\*
end     \*\/



%%
\/\/(.*) ;
\/\*(.*\n)*.*\*\/  {};

int     return INT;
float   return FLOAT;
char    return CHAR;
bool    return BOOL;
double  return DOUBLE;
static  return STATIC;
main    return MAIN;
if      return IF;
else    return ELSE;
```

```
for       return FOR;
while     return WHILE;
do        return DO;
break     return BREAK;
#include return INCLUDE;
"<"(.+)".h>" return HEADER;
{id}      return ID;
{number}          return VNUM;
{strlit}  return STRLIT;

"<"       return LT;
">"       return GT;
">="      return GTE;
"<="      return LTE;
"=="      return EQ;
"!="      return NE;
"++"      return INC;
"--"      return DEC;
"||"      return OR;
"&&"      return AND;
"!"       return LNOT;
"("       return SCOMB;
")"       return ECOMB;
"["       return SSQB;
"]"       return ESQB;
"{"       return SCURB;
"}"       return ECURB;
\r        ;
[' '|'\t'] ;
\n        ++yylineno;
.         return *yytext;

%%
int yywrap()
{

}
```

Parser.y

```
%{

#include<stdio.h>
#include<stdlib.h>
int yylex();
void yyerror(char* s);
extern int yylineno;
extern char *yytext;
%}
%token INT FLOAT DOUBLE CHAR STATIC ID INCLUDE HEADER MAIN DO WHILE IF ELSE FOR
BOOL BREAK INC DEC STRLIT VNUM LT GT GTE LTE EQ NE OR AND LNOT SCOMB ECOMB
SSQB ESQB SCURB ECURB
%start P
%%


P : S {printf("Valid Syntax\n");YYACCEPT;}
  ;
S : INCLUDE HEADER S
  | STATIC S
  | MAINF S
  | DECLR ';' S
  | ASSGN ';' S
  |
  ;

TYPE : INT | FLOAT | CHAR | BOOL | DOUBLE
     ;

DECLR : TYPE List_Var | s
      ;

List_Var : List_Var ',' ID | ID
         ;

ASSGN : TYPE ID '=' EXPR | ID '=' EXPR | STRLIT
      ;

EXPR : EXPR RELOP E | E | ID INC | ID DEC | LNOT ID|S
     ;

RELOP : GTE|LTE|EQ|NE|OR|AND|LT|GT
      ;

E : E'+'T|E'-'T|T
  ;

T : T'*'F|T'/'F|F
  ;


F : SCOMB EXPR ECOMB | ID | VNUM
  ;

MAINF : TYPE MAIN SCOMB Empty_ListVar ECOMB SCURB Stmt ECURB
```

```
    ;

Empty_ListVar : List_Var
              |
           ;

Stmt : SingleStmt Stmt | Block Stmt | BREAK
    |
    ;

Ifelstmt : SingleStmt Stmt | Block Stmt
          ;


SingleStmt : DECLR ';' | ASSGN ';' | IF SCOMB COND ECOMB Ifelstmt | IF SCOMB COND
ECOMB Ifelstmt ELSE Ifelstmt | LOOP | DO Block WHILE COND ';'
         ;

Block : SCURB Stmt ECURB
     ;

LOOP : WHILE SCOMB COND ECOMB LOOP2
     | FOR SCOMB COND ECOMB LOOP2
     ;

COND : EXPR | ASSGN
    ;

LOOP2 : SCURB Stmt ECURB
     |
     ;

s : error;
%%
void yyerror(char* s)
{
printf("Error:%s,line number:%d,token:%s\n",s,yylineno,yytext);
}

int main()
{
yyparse();

}
```

TEAM NO:22
SRN1:PES2UG22CS275 KUSHAGRA AGARWAL
SRN2:PES2UG22CS261 KEERTHANA
SRN3:PES2UG22CS910 ARJUN N R
SRN4:PES2UG22CS133 Bhuvan CV

5. Answer the following:

1. Total Number of Parameters in Each CNN Architecture

The total number of parameters in different CNN architectures for MNIST are:

- LeNet-5: ≈ 61,706
- AlexNet: ≈ 5,038,346
- ResNet-18: Significantly larger than both LeNet-5 and AlexNet, likely in the order of millions or tens of millions.

2. Correlation Between Model Parameters and Accuracy on MNIST & CIFAR-10

In general, a higher number of parameters allows a model to learn more complex patterns, potentially improving accuracy.

- LeNet-5: Lowest capacity, performs well on MNIST but struggles with CIFAR-10.
- AlexNet: Higher capacity than LeNet-5, expected to outperform it on both datasets, especially CIFAR-10.
- ResNet-18: Highest capacity, likely to achieve the best accuracy on both datasets, particularly CIFAR-10.

However, increased parameters do not guarantee higher accuracy. Performance depends on dataset size, training process, and architecture efficiency.

3. Impact of Dataset Complexity (Grayscale vs. RGB) on Accuracy and Training Time

Impact on Accuracy:

- RGB Increases Complexity: CIFAR-10 (RGB) is more complex than MNIST (grayscale).
- Initially Lower Accuracy: Models may initially struggle with CIFAR-10 due to increased variability.
- Higher Accuracy With Advanced Models: Deeper architectures like AlexNet and ResNet-18 can leverage color information for better accuracy.

Impact on Training Time:

- Larger Input Size: RGB images are three times larger than grayscale, increasing computational load.
- More Complex Feature Learning: Extracting useful features from color images requires more processing.
- Longer Training Duration: CIFAR-10 requires more time for training compared to MNIST.

4. Trade-Offs Between Accuracy and Computational Efficiency

- Higher Accuracy Requires More Complexity: More parameters lead to better accuracy but demand greater computational resources.
- Simpler Models Are More Efficient: Models with fewer parameters are computationally efficient but may have lower accuracy.

The choice depends on application needs—critical tasks may prioritize accuracy, while resource-limited environments may favor efficiency. Finding the right balance is essential.