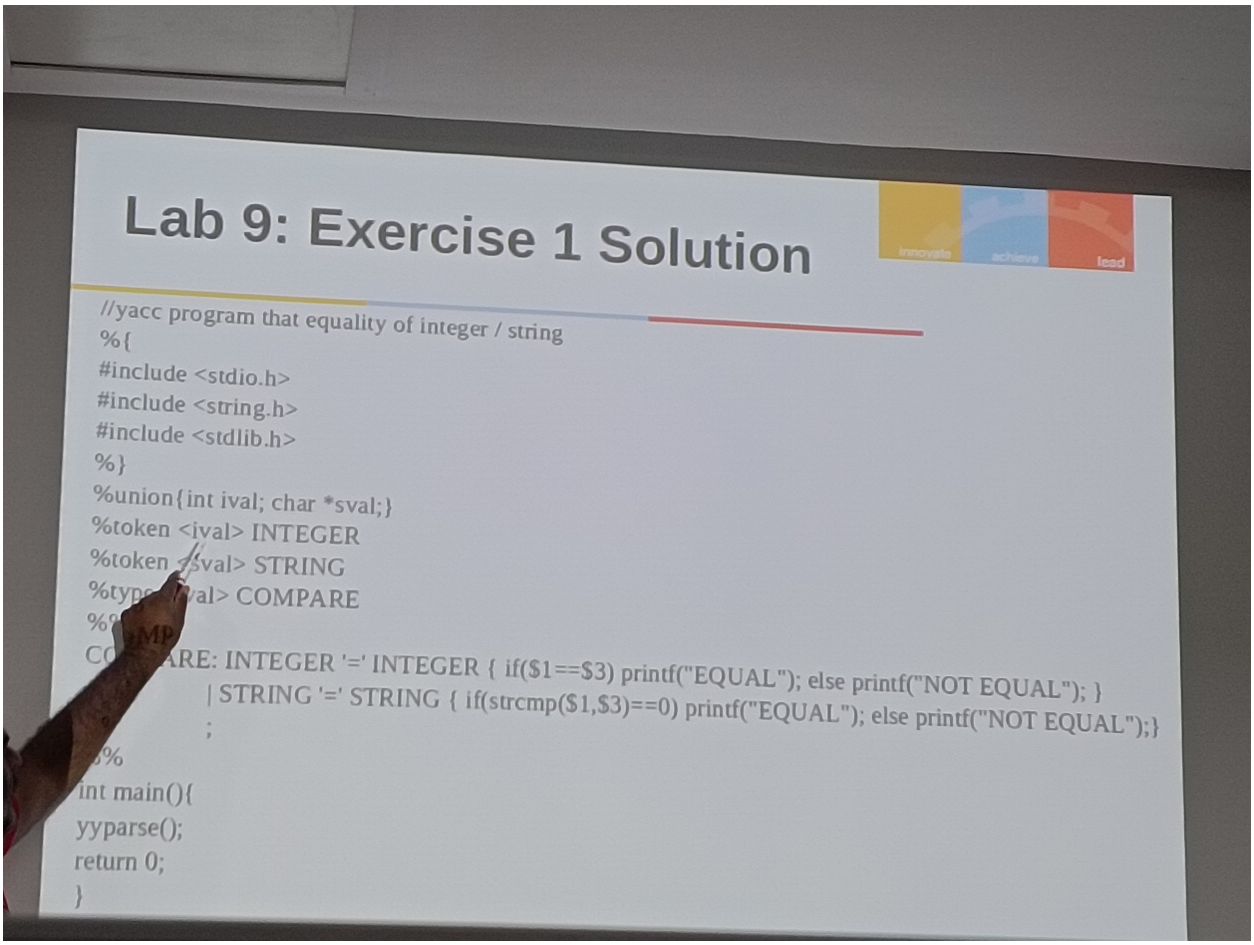
Lab 9

Write a parser program that checks for the equality of integer / string



lex

```
%{
#include "y.tab.h"
#include<math.h>
#include<string.h>
#include<stdlib.h>
%}

%%
```

```
[0-9]+ {yylval.ival=atoi(yytext); return INTEGER;}
[a-zA-Z]+ {yylval.sval=malloc(sizeof(char)*10); strcpy(yylval.sv
"=" {return EQ;}
\n return NL;
%%
```

yacc

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
%}

%union {int ival; char *sval;}
%token EQ NL
%token <ival> INTEGER
%token <sval> STRING
%type <ival> compare

%%
stmt    : compare NL {return 1;}
compare: INTEGER EQ INTEGER {if ($1=$3) printf("EQUAL'); else pr
    | STRING EQ STRING {if(strcmp($1, $3)==0) printf("EQUAL"); e
    ;
%%


int main()
{
    yyparse();
    return 0;
}

int yyerror()
{
```

```
    printf("Invalid\n");
    return 0;
}
```

Exercise 2

Exercise 2: Write Lex and Yacc programs to validate the following assignment statements.
id=id;
id=number;
Note that ID can start with a-z or A-Z only, and followed by any combination of a-z; A-Z; 0-9. NUMBER is [0-9]+
If the input is : sum=total; // print that it is a valid assignment statement.
If the input is temp =123; // // print that it is a valid assignment statement.
If input is 124=temp; //print invalid statement
12=12; //print invalid statement
So on...

lex

```
%{
#include "y.tab.h"
#include<math.h>
#include<string.h>
#include<stdlib.h>
%}

%%
[0-9]+ {return NUM;}
[a-zA-Z][A-Za-z0-9]+ {return ID;}
"=" return EQ;
";" return SC;
\n return NL;
```

```
%%
```

yacc

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
%}

%token NUM NL ID EQ SC

%%
stmt    : assign NL {return 1;}
assign  : ID EQ ID SC {}
    | ID EQ NUM SC {}
    ;
%%


int main()
{
    yyparse();
    return 0;
}

int yyerror()
{
    printf("Invalid\n");
    return 1;
}
```

Exercise 3

Exercise 3: Write Yacc and Lex program parses declaration int and float variables like-
int sum;
int num, sum, total;
float sum;
float num, sum, total;
etc.

Exercise 4

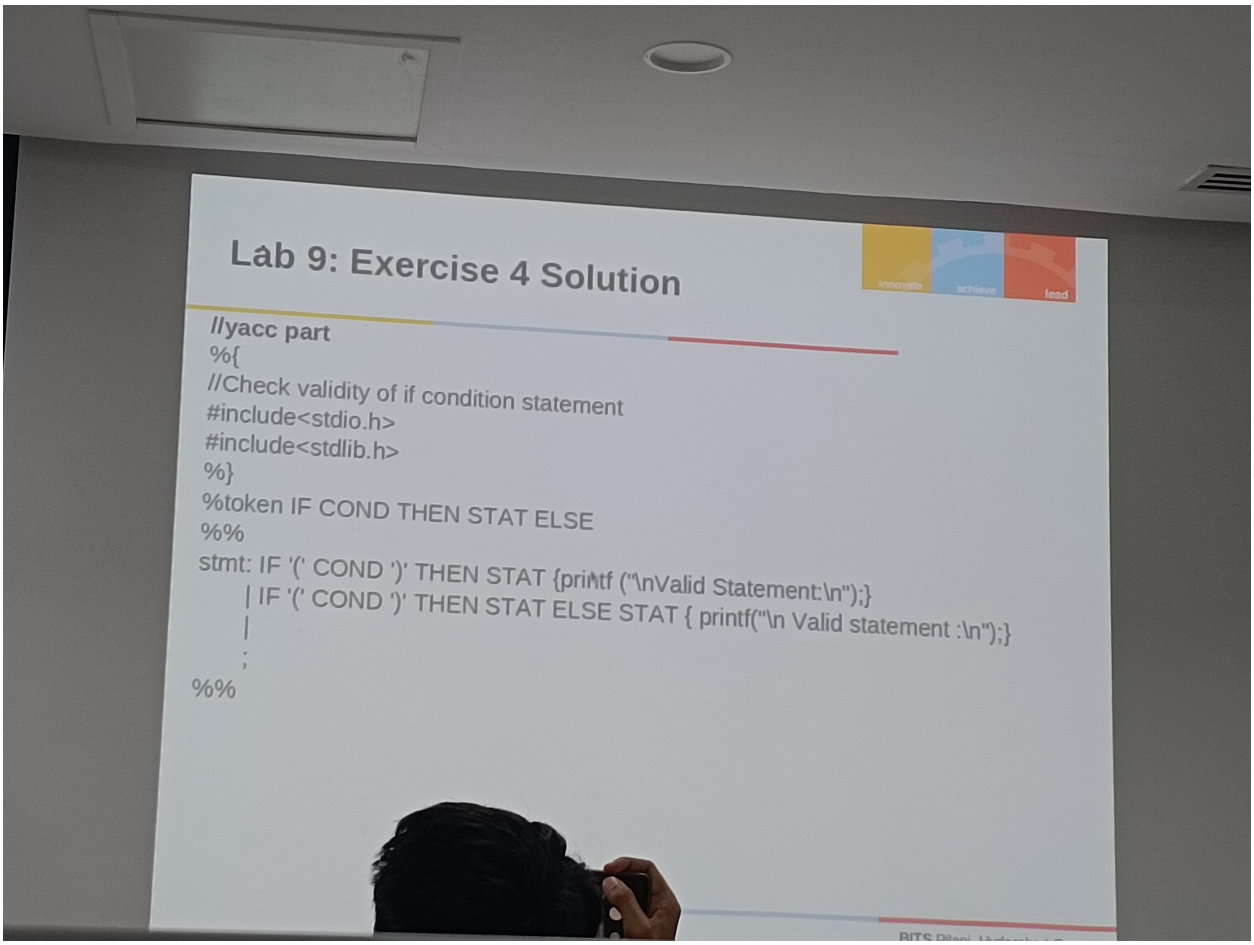
Exercise 4: Write a parser to validate the following IF statements.
IF (COND) THEN STMT
OR
IF (COND) THEN STMT ELSE STMT
COND can be [a-z]+
STMT can be [a-z}+
Clue: use the concept of states already learned. No nesting of Ifs

Idea:

- Create (for open brace
-

Lab 9: Exercise 4 Solution

```
//lex part
%{
#include "y.tab.h"
%}
CHAR [a-zA-Z0-9]
%x CONDSTART
%%
<*>[ ] ;
<*>[ \t]+ ;
<*>[\n] {return 0;}
"if" {return IF;}
"else" {return ELSE;}
"then" {return THEN;}
\( { BEGIN (CONDSTART); return('(');}
<CONDSTART>{CHAR}+ return COND;
<CONDSTART>\) {BEGIN (INITIAL); return(')');}
{CHAR}+ return STAT;
%%
```



Exercise 5

Write a program to count number of 1s in a binary string generated by CFG:

$$N \to L, L \to LB, L \to B, B \to 0, B \to 1$$

```
%{
#include "y.tab.h"
#include<math.h>
#include<string.h>
#include<stdlib.h>
%}

%%
```

```
[0] {return ZERO; }
[1] {return ONE; }
\n return NL;
%%
```

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
%}

%token ZERO ONE NL

%%
stmt    : l NL { printf("Number of 1s is %d\n", $1); return 1;}
    ;
l   : l b {$$=$1+$2; }
    | b {$$=$1;}
    ;
b   : ZERO {$$=0;}
    | ONE {$$=1;}
    ;
%%


int main()
{
    yyparse();
    return 0;
}

int yyerror()
{
    printf("Invalid\n");
```

```
    return 1;
}
```

EXERCISE 6

Write a program to convert the binary string generated by CFG to an equivalent decimal:

$$N \rightarrow L, L \rightarrow LB, L \rightarrow B, B \rightarrow 0, B \rightarrow 1$$

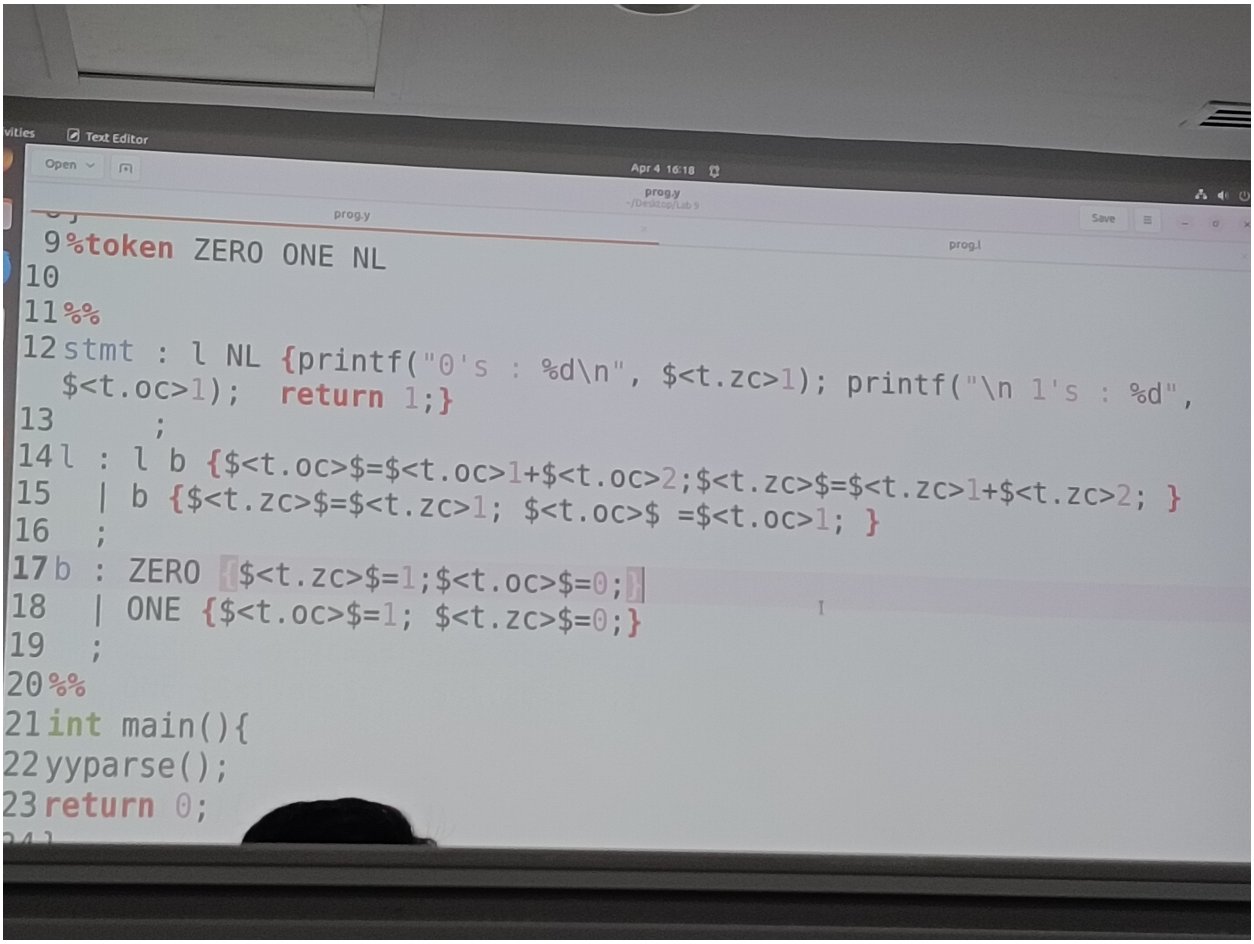
Lex Program is the same

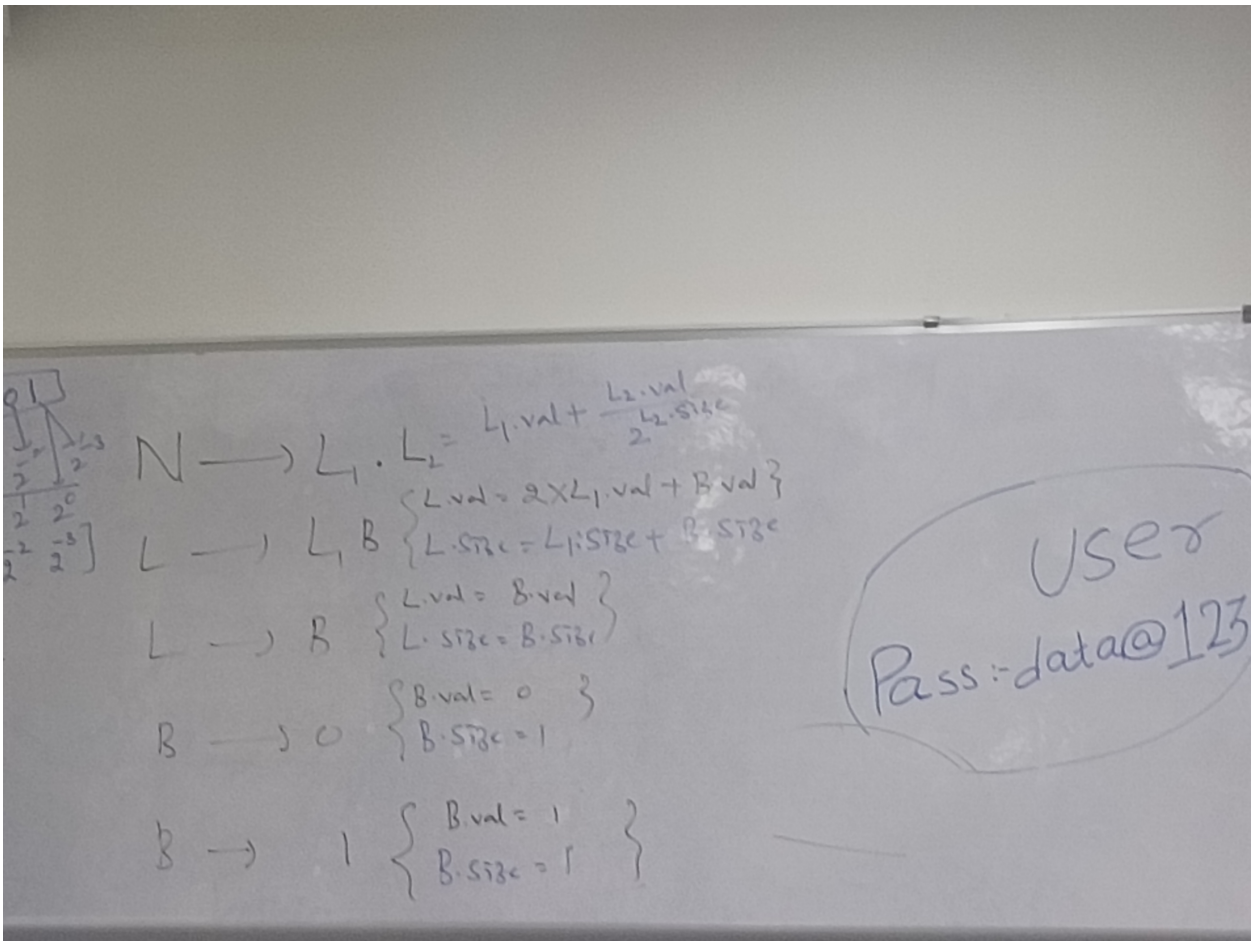
Yacc

```
%{
#include<stdio.h>
#include<string.h>
#include<stdlib.h>
%}

%token ZERO ONE NL

%%
stmt    : l NL { printf("Decimal eqvt is %d\n", $1); return 1;}
    ;
l   : l b {$$=2*$1+$2; }
    | b {$$=$1;}
    ;
b   : ZERO {$$=0;}
    | ONE {$$=1;}
    ;
%%


int main()
{
```

```
    yyparse();
    return 0;
}

int yyerror()
{
    printf("Invalid\n");
    return 1;
}
```

Exercise 7





Homework 1

Home Work 1: Write a program to print the fractional decimal number equivalent to the binary fractional number represented by the binary string that can be generated by CFG:
N→L '.' L
L→LB
L→B
B→1
B → 0

Homework 2

- Distinguish between type and return type

•