EVENT 2

YACC PROOGRAMS!!

→ Create lex file (command: vi file.l)
→ Create Yacc file (command: vi file.y)
→ **NOTE:** Lex and yacc file should have the same file name!!
→ Compilation: (You can compile either lex or yacc file first)
  Use these Commands for Compilation:

```
lex file.l

yacc -d file.y

gcc lex.yy.c y.tab.c
```

1. Yacc Program to check for valid Expression

LEX Code:

```
%{
#include "y.tab.h"
%}

%%
[0-9]+ {  yylval = atoi(yytext); return num ; }
[_a-zA-Z] {return id;}
[\t ];
.|\n { return yytext[0]; }
%%

int yywrap(){ return 1; }
```

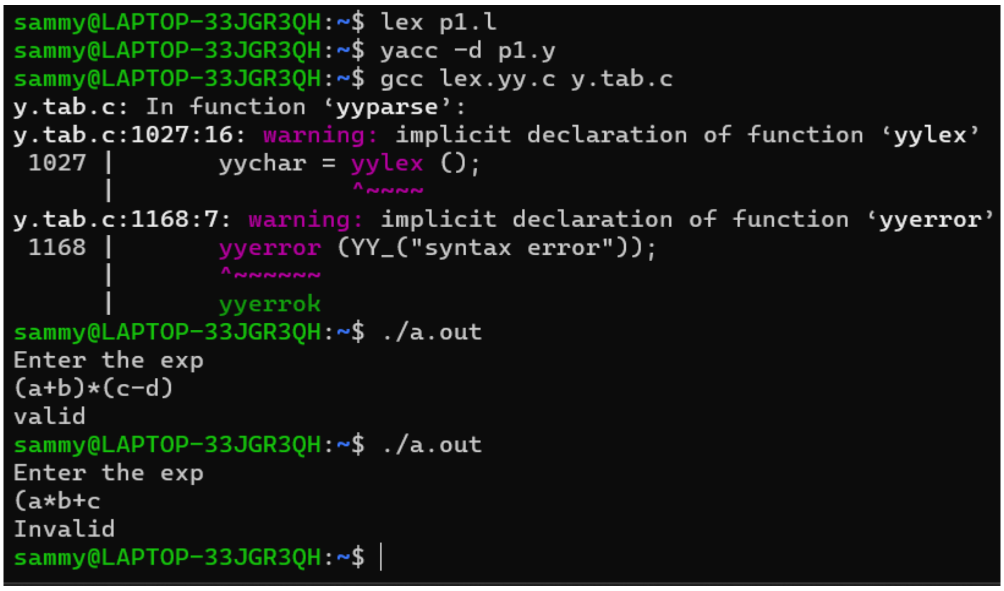
YACC Code:

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

```
%token id num
%left '+''-'
%left '*'
%left '/'
%%
stat:exp'\n' {printf("valid\n");exit(0);}
   ;
exp:exp'+'exp|exp'-'exp|exp'*'exp|exp'/'exp|'('exp')'|id|num;
%%

int main()
{
printf("Enter the exp\n");
yyparse();
return 0;
}
int yyerror()
{
printf("Invalid\n");
exit(0);
}
```

SAMPLE OUTPUT (Ignore the Warning)



```
sammy@LAPTOP-33JGR3QH:~$ lex p1.l
sammy@LAPTOP-33JGR3QH:~$ yacc -d p1.y
sammy@LAPTOP-33JGR3QH:~$ gcc lex.yy.c y.tab.c
y.tab.c: In function ‘yyparse’:
y.tab.c:1027:16: warning: implicit declaration of function ‘yylex’
 1027 |       yychar = yylex ();
      |                ^~~~~
y.tab.c:1168:7: warning: implicit declaration of function ‘yyerror’
 1168 |       yyerror (YY_("syntax error"));
      |       ^~~~~~~
      |       yyerrok
sammy@LAPTOP-33JGR3QH:~$ ./a.out
Enter the exp
(a+b)*(c-d)
valid
sammy@LAPTOP-33JGR3QH:~$ ./a.out
Enter the exp
(a*b+c
Invalid
sammy@LAPTOP-33JGR3QH:~$
```

2. Yacc program to check for the correctness of valid identifier

LEX Code:

```
%{
#include "y.tab.h"
%}

%%
[0-9]+ { return digit ; }
[_] { return under_score; }
[a-zA-Z] { return letter; }
.|\n { return yytext[0]; }
%%

int yywrap(){ return 1 ; }
```

YACC Code:

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

%token digit letter under_score

%%
id: exp'\n' { printf( "Valid identifier\n") ; exit(0) ; }
  ;
exp: letter x
   | under_score x
   ;
```

```
x : digit x

  | letter x
  | under_score x
  |
  ;
%%

int main()
{

  printf("Enter variable name \n");
  yyparse();
  return 0;
}
int yyerror()
{
  printf("invalid identifier\n");
  exit(0);
  return 0;
}
```

3. Yacc program to Evaluation of valid mathematical expression

LEX Code:

```
%{
#include "y.tab.h"
%}

%%
[0-9]+ {  yylval = atoi(yytext); return NUMBER ; }
[\t ];
.|\n { return yytext[0]; }
%%

int yywrap(){ return 1; }
```

YACC Code:

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

%token NUMBER
%left '+' '-'
%left '*'
%left '/'

%%
stmt:E'\n'{ printf("Valid Expression evaluates to %d \n", $$ ); exit(0); }
;
E: E '+' E { $$ = $1 + $3; }
```

```
 | E '-' E { $$ = $1 - $3; }
 | E '*' E { $$ = $1 *  $3;  }
 | E '/' E { if($3==0){ printf("Division by 0 \n"); exit(1); } $$ = $1 / $3 ; }
 |'(' E ')' { $$ = $2 ; }
 | NUMBER { $$ = $1; }
 ;
%%

int main(){
  printf("Enter the Expression in terms of integers\n");
  yyparse();
  return 0;
}

int yyerror(){ printf("Invalid Expression\n"); exit(0); return 0; }
```

4. Yacc program to check for the pattern for A^nB^n

LEX Code:

```
%{
#include "y.tab.h"
%}


%%
[aA] { return A; }
[bB] { return B;}
\n { return '\n'; }
. { return yytext[0];}
%%



int yywrap(){ return 1 ; }
```

YACC Code:

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



%token A B
```

```
%%
stmt:s'\n' { printf("valid string\n"); exit(0);}
    ;
s:A s B|
 ;

%%



int main(){
  printf("Enter the string\n");
  yyparse();
  return 0;
}

int yyerror(char *msg)
{
  printf("Invalid String\n");
  exit(0);
}
```

5. Yacc program to implement a Calculator and recognize a valid Arithmetic expression

LEX Code:

```
%{
/* Definition section */
#include<stdio.h>
#include "y.tab.h"
extern int yylval;
%}

/* Rule Section */
%%
[0-9]+ {
                yylval=atoi(yytext);
                return NUMBER;
       }
[\t] ;

[\n] return 0;

. return yytext[0];

%%

int yywrap()
{
return 1;
}
```

YACC Code:

```
%{
/* Definition section */
#include<stdio.h>
int flag=0;
%}

%token NUMBER

%left '+' '-'

%left '*' '/' '%'

%left '(' ')'

/* Rule Section */
%%

ArithmeticExpression: E{

                        printf("\nResult=%d\n", $$);

                        return 0;

                        };
E:E'+'E {$$=$1+$3;}

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

|E'*'E {$$=$1*$3;}
```

```
|E'/'E {$$=$1/$3;}

|E'%'E {$$=$1%$3;}

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

| NUMBER {$$=$1;}

;

%%

//driver code
void main()
{
printf("\nEnter Any Arithmetic Expression which can have operations Addition, Subtraction,
Multiplication, Division, Modulus and Round brackets:\n");
yyparse();
if(flag==0)
printf("\nEntered arithmetic expression is Valid\n\n");
}

void yyerror()
{
printf("\nEntered arithmetic expression is Invalid\n\n");
flag=1;
}
```