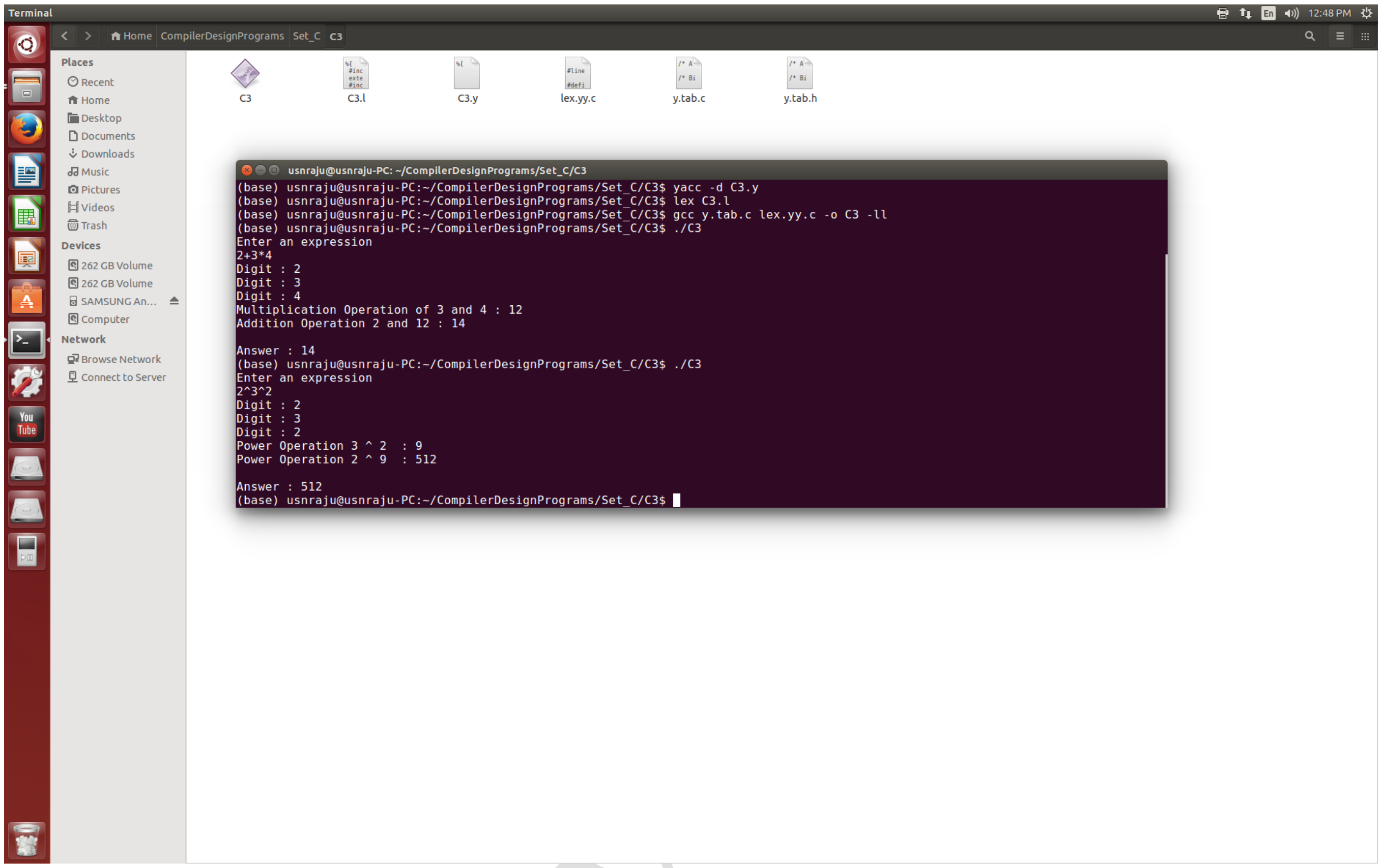
/* C3: Use YACC to implement: Expression values evaluation (Desktop calculator). */

File C3.y

```
/* definition section*/
%{
        #include <stdio.h>
        #include <ctype.h>
        int x[5],y[5],k,j[5],a[5][10],e,w;
%}

// creating tokens whose values are given by lex
%token digit

// following a grammer rule which is printing the digit first
then solving
// the expression of additiion ,subtraction,multiplication,and
power .
%%

S : E { printf("\nAnswer : %d\n",$1); }
  ;

E : T { x[e]=$1; } E1 { $$=x[e];  }
  ;

E1 : '+' T { w=x[e]; x[e]=x[e]+$2; printf("Addition Operation %d
and %d : %d\n",w,$2,x[e]);  } E1 { $$=x[e]; }
   | '-' T { w=x[e]; x[e]=x[e]-$2; printf("Subtraction Operation
%d and %d : %d\n",w,$2,x[e]); } E1 { $$=x[e]; }
   | { $$=x[e]; }
   ;

T : Z { y[e]=$1; } T1 { $$=y[e]; }
  ;

T1 : '*' Z { w=y[e];  y[e]=y[e]*$2; printf("Multiplication
Operation of %d and %d : %d\n",w,$2,y[e]);  } T1 { $$=y[e]; }
   | { $$=y[e]; }
   ;

Z : F { a[e][j[e]++]=$1; } Z1 { $$=$3; }
  ;
Z1 : '^' Z { $$=$2; }
   | { for(k=j[e]-1;k>0;k--) { w=a[e][k-1];  a[e][k-
1]=powr(a[e][k-1],a[e][k]); printf("Power Operation %d ^ %d  :
%d\n",w,a[e][k],a[e][k-1]);  } $$=a[e][0]; j[e]=0; }
   ;

F : digit { $$=$1; printf("Digit : %d\n",$1); }
  | '(' { e++; } E { e--; } ')' { $$=$3; }
  ;
%%

int main()
{
     //initializing all the variables to zero
        for(e=0;e<5;e++) { x[e]=y[e]=0; j[e]=0; }
        e=0;
     // takes input as a expression
        printf("Enter an expression\n");
        yyparse();
        return 0;
}
// if any error yyerror will be called
yyerror()
{
   printf("NITW Error");
}
// when the input is finished yywrap is called to exit the code
int yywrap()
{
  return 1;
}
// power function to calculate m ^ n
int powr(int m,int n)
{
        int ans=1;
        while(n) { ans=ans*m; n--; }
        return ans;
}
```

File C3.l

```
/* definitions */
%{
// including required header files
 #include "y.tab.h"
 #include <stdlib.h>
// declaring a external variable yylval
 extern int yylval;
%}
%%
//If the token is an Integer number,then return it's value.
[0-9]+ {yylval=atoi(yytext);return digit;}
//If the token is space or tab,then just ignore it.
[\t]   ;
//If the token is new line,return 0.
[\n]   return 0;
//For any other token, return the first character read since the
last match.
.      return yytext[0];
%%
```



```
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C3$ yacc -d C3.y
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C3$ lex C3.l
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C3$ gcc y.tab.c lex.yy.c -o C3 -ll
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C3$ ./C3
Enter an expression
2+3*4
Digit : 2
Digit : 3
Digit : 4
Multiplication Operation of 3 and 4 : 12
Addition Operation 2 and 12 : 14

Answer : 14
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C3$ ./C3
Enter an expression
2^3^2
Digit : 2
Digit : 3
Digit : 2
Power Operation 3 ^ 2  : 9
Power Operation 2 ^ 9  : 512

Answer : 512
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C3$
```