/* C10: Write a YACC "desk calculator" program that will evaluate Boolean expressions, consists of $\neg, \wedge, \vee, \rightarrow$ *and* $\leftrightarrow$ where, $\neg$ is highest precedence and $\leftrightarrow$ is with lowest precedence.

INPUT : $T \wedge F \rightarrow (T \vee F)$ **OUTPUT**: T */

File: C10.y

```
%{
   /* Definition section */
   #include <ctype.h>
   #include<stdio.h>
   #include<stdlib.h>
%}
%token digit
/* Rule Section */
%%

/* Initialize the parsing and print final result. */
S: E {if ($1==1)
       printf("\n\nFinla Result is : T\n\n");
      else
       printf("\n\nFinla Result is : F\n\n"); }
 ;

/* Double implies has lowest precedence. */
/* Split the expression into two halves and reduce the LHS and RHS and
finally compute the result of the double implies. */
/* If double implies isn't found, go to the operator with the next
higer precedence. */
E:  E '<''-''>' T {   if ($1==0 && $5==0){$$=1;}
                      if ($1==0 && $5==1){$$=0;}
                      if ($1==1 && $5==0){$$=0;}
                      if ($1==1 && $5==1){$$=1;}
                      printf("<-> is computed:Result is %d\n",$$);
                  }
 |  T {$$=$1;}
 ;

/* Implies has second lowest precedence. */
/* Split the expression into two halves and reduce the LHS and RHS and
finally compute the result of the implies. */
/* If implies isn't found, go to the operator with the next higer
precedence. */
T:  T '-''>' F  {   if ($1==0 && $4==0){$$=1;}
                    if ($1==0 && $4==1){$$=1;}
                    if ($1==1 && $4==0){$$=0;}
                    if ($1==1 && $4==1){$$=1;}
                    printf("-> is computed:Result is %d\n",$$);
                }
 |  F {$$=$1;}
 ;

/* OR has third lowest precedence. */
/* Split the expression into two halves and reduce the LHS and RHS and
finally compute the result of the OR logic. */
/* If OR operator isn't found, go to the operator with the next higer
precedence. */
F:  F 'v' M { $$=$1+$3;
              printf("v is computed:Result is %d\n",$$);
            }
  | F 'V' M { $$=$1+$3;
              printf("v is computed:Result is %d\n",$$);
            }

 |  M {$$=$1;}
 ;

/* AND has Third highest precedence. */
/* Split the expression into two halves and reduce the LHS and RHS and
finally compute the result of the AND logic. */
/* If AND operator isn't found, go to the operator with the next higer
precedence. */
M:  M '^' N { $$=$1*$3;
              printf("^ is computed:Result is %d\n",$$);
            }
 |  N {$$=$1;}
 ;

/* NOT has second highest precedence. Highest as per the logic gates.
*/
/* Evaluate using the logic gate of NOT and search if there are
brackets or T/F. */
/* If OR operator isn't found, go to the operator with the next higer
precedence. */
N:  '!' N  { if ($2==1){$$=0;}
             if ($2==0){$$=1;}
             printf("! is computed:Result is %d\n",$$);
           }
 |  P  {$$=$1;}
 ;

/* Brackets have the highest precedence. */
/* If brackets are found, first reduce the expression inside the
brackets. */
/* If a T/F is found, return the corresponding 0/1 value. */
P: '(' E ')'  {$$=$2;}
  | digit {if ($1=='T'){printf("%c",$1);$$=1;}
           if ($1=='F'){printf("%c",$1);$$=0;} }



  ;
```

```
%%

//driver code
int main()
{
    printf("Enter infix expression:  ");
    yyparse();
}
yyerror()
{
   printf("Error");
}
```

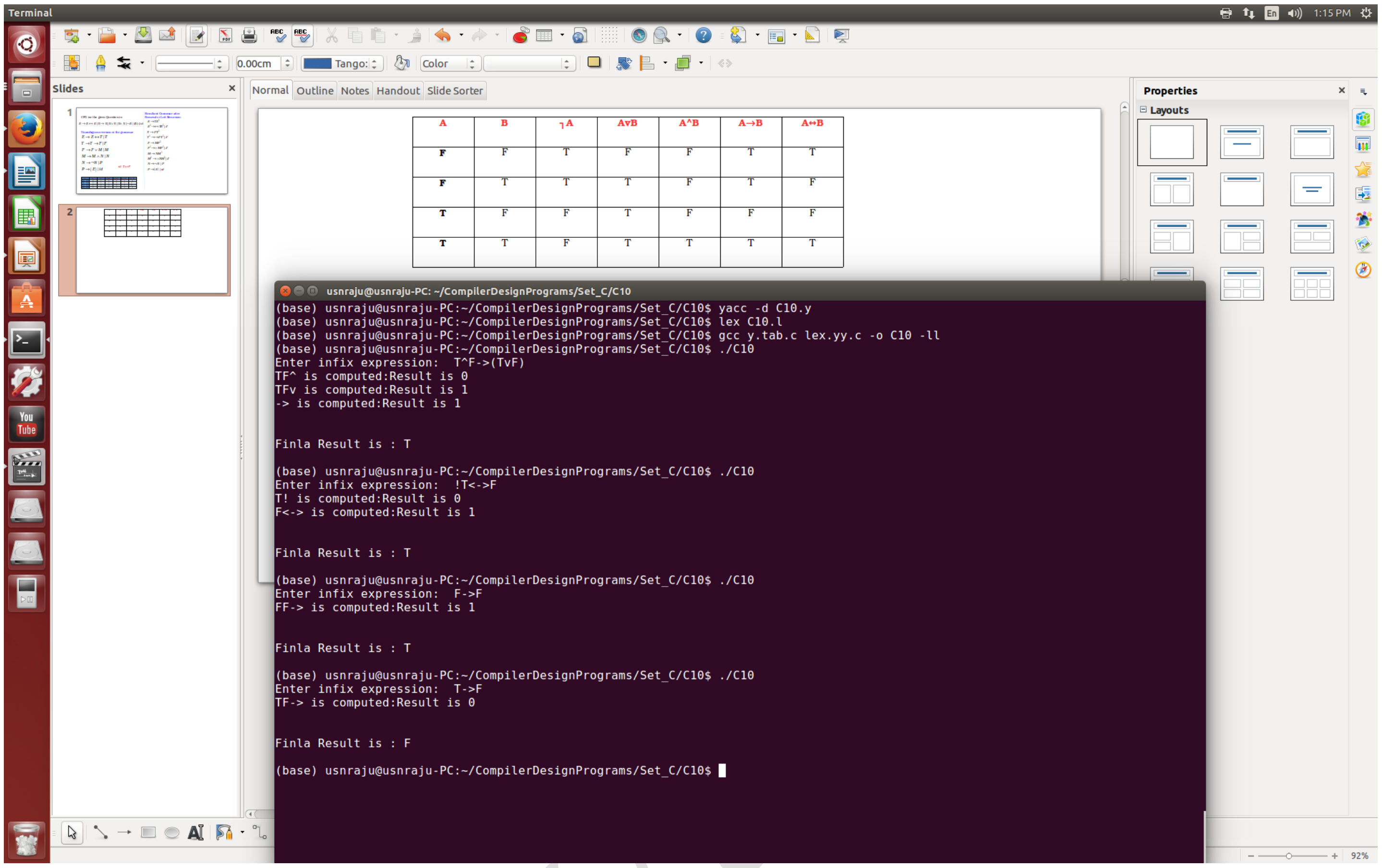
File: C10.l

```
%{
        #include "y.tab.h"
        extern int yylval;
        #include <stdlib.h>
%}

/*
        Rules:
                If 'T' is matched, send it as a token.
                If 'F' is matched, send it as a token.
                If a tab is mathced, do nothing.
                If a new line character is matched, end the parsing.
                For any other word or character, send the first
character as the token.
*/

%%

[T] {  yylval=yytext[0]; return digit;}
[F] {  yylval=yytext[0]; return digit;}
[\t]  ;
[\n]  return 0;
 .    return yytext[0];
%%
```



A	B	┐A	AvB	A^B	A→B	A↔B
F	F	T	F	F	T	T
F	T	T	T	F	T	F
T	F	F	T	F	F	F
T	T	F	T	T	T	T

usnraju@usnraju-PC: ~/CompilerDesignPrograms/Set_C/C10

```
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C10$ yacc -d C10.y
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C10$ lex C10.l
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C10$ gcc y.tab.c lex.yy.c -o C10 -ll
(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C10$ ./C10
Enter infix expression:  T^F->(TvF)
TF^ is computed:Result is 0
TFv is computed:Result is 1
-> is computed:Result is 1


Finla Result is : T

(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C10$ ./C10
Enter infix expression:  !T<->F
T! is computed:Result is 0
F<-> is computed:Result is 1


Finla Result is : T

(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C10$ ./C10
Enter infix expression:  F->F
FF-> is computed:Result is 1


Finla Result is : T

(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C10$ ./C10
Enter infix expression:  T->F
TF-> is computed:Result is 0


Finla Result is : F

(base) usnraju@usnraju-PC:~/CompilerDesignPrograms/Set_C/C10$ 
```