# Lex and Yacc for ‘if’ conditional statements

## Lex:

```
/ Lex rules for if conditional statement

ALPHA [A-Za-z]
DIGIT [0-9]
%%
if return IF;
then return THEN;
else return ELSE;
{ALPHA}({ALPHA}|{DIGIT})* return ID;
{DIGIT}+ {yylval=atoi(yytext); return NUM;}
[ \t] ;
\n yyterminate();
. return yytext[0];
%%
```

## Yacc:

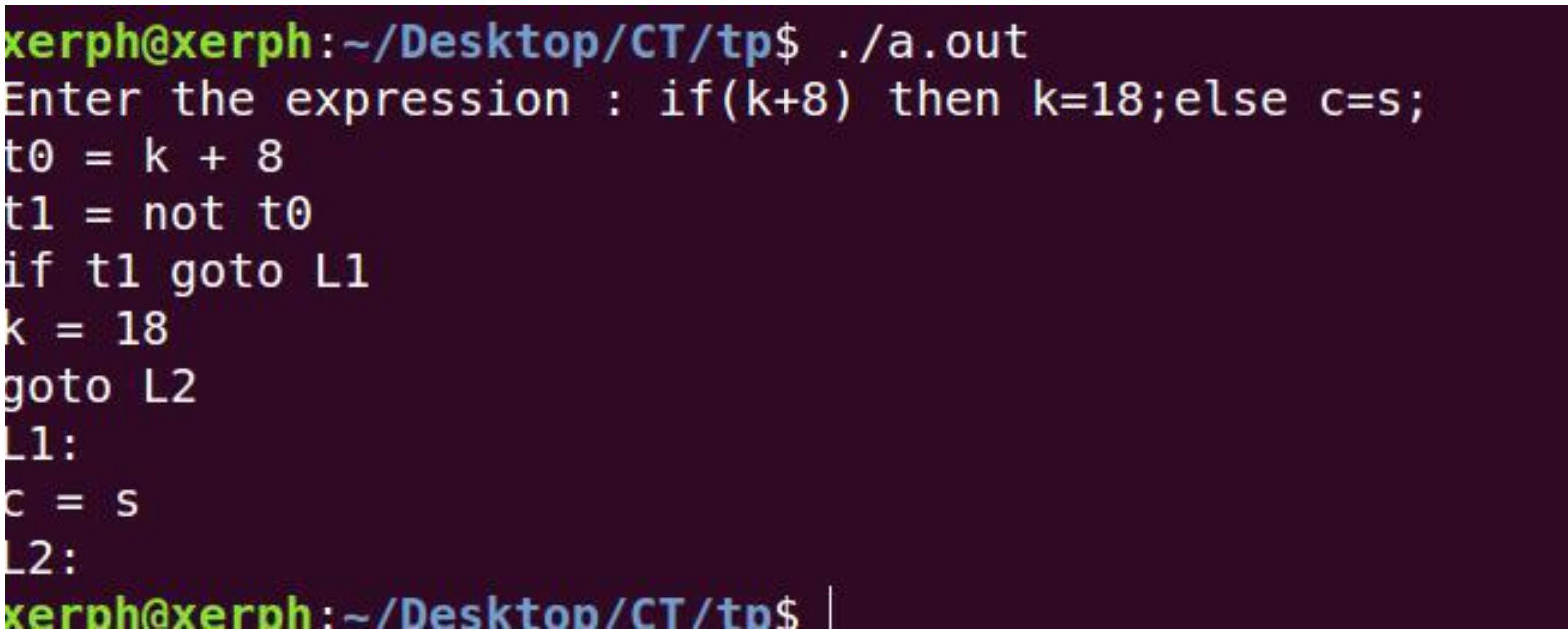
```
/ Yacc grammar for if conditional statement


%token ID NUM IF THEN ELSE
```

```
%right '='
%left '+' '-'
%left '*' '/'

%%
S : IF '(' E ')'{lab1();} THEN E ';'{lab2();} ELSE E ';'{lab3();}
;
E :V '='{push();} E{codegen_assign();}
| E '+'{push();} E{codegen();}
| E '-'{push();} E{codegen();}
| E '*'{push();} E{codegen();}
| E '/'{push();} E{codegen();}
| '(' E ')'

| V
| NUM{push();}
;
V : ID {push();}
;
%%
#include "lex.yy.c"
#include<ctype.h>
char st[100][10];
int top=0;
char i_[2]="0";
char temp[2]="t";
int label[20]; int
lnum=0;
int ltop=0;
main()
 {
```

```
 printf("Enter the expression : ");
yyparse();
}
push()
 {
  strcpy(st[++top],yytext);
 }
codegen()
 {
 strcpy(temp,"t");
 strcat(temp,i_);
  printf("%s = %s %s %s\n",temp,st[top-2],st[top-
1],st [top]);
  top-=2;
 strcpy(st[top],temp);
 i_[0]++;
 }

codegen_assign()
 {
 printf("%s = %s\n",st[top-2],st[top]);
top-=2;
}
lab1()
{
 lnum++;
 strcpy(temp,"t");
 strcat(temp,i_);
 printf("%s = not %s\n",temp,st[top]);
printf("if %s goto L%d\n",temp,lnum);
i_[0]++;
```

```
label[++ltop]=lnum;
}
lab2()
{
int x;
lnum++;
x=label[ltop--];
printf("goto L%d\n",lnum);
printf("L%d: \n",x);
label[++ltop]=lnum;
}
lab3()
{
int y;
y=label[ltop--];
printf("L%d: \n",y);
}
int yyerror()
{
}
int yywrap()
{
return 1;
}
```

**OUTPUT:**


xerph@xerph:~/Desktop/CT/tp$ ./a.out
Enter the expression : if(k+8) then k=18;else c=s;
t0 = k + 8
t1 = not t0
if t1 goto L1
k = 18
goto L2
L1:
c = s
L2:
xerph@xerph:~/Desktop/CT/tp$