```c
# include "stdio.h"
# define U(x) ((x)&0377)
# define NLSTATE yyprevious=YYNEWLINE
# define BEGIN yybgin = yysvec + 1 +
# define INITIAL 0
# define YYLERR yysvec
# define YYSTATE (yyestate-yysvec-1)
# define YYOPTIM 1
# define YYLMAX 200
# define output(c) putc(c,yyout)
# define input() (((yytchar=yysptr>yysbuf?U(*--yysptr):getc(yyin))==10?(yylineno++,yytch
ar):yytchar)==EOF?0:yytchar)
# define unput(c) {yytchar= (c);if(yytchar=='\n')yylineno--;*yysptr++=yytchar;}
# define yymore() (yymorfg=1)
# define ECHO fprintf(yyout, "%s",yytext)
# define REJECT { nstr = yyreject(); goto yyfussy;}
int yyleng; extern char yytext[];
int yymorfg;
extern char *yysptr, yysbuf[];
int yytchar;
FILE *yyin ={stdin}, *yyout ={stdout};
extern int yylineno;
struct yysvf {
        struct yywork *yystoff;
        struct yysvf *yyother;
        int *yystops;};
struct yysvf *yyestate;
extern struct yysvf yysvec[], *yybgin;
#include <stdio.h>
#include "tokdefs.one.h"
# define YYNEWLINE 10
yylex(){
int nstr; extern int yyprevious;
/* In this section any lines beginning with a blank are  */
        /* copied into lex.yy.c                                  */

 printf("*****\n");
while((nstr = yylook()) >= 0)
yyfussy: switch(nstr){
case 0:
if(yywrap()) return(0); break;
case 1:
{ /* no action and no return */ }
break;
case 2:
{ return(IFtok); }
break;
case 3:
{ return(THENtok); }
break;
case 4:
{ return(ELSEtok); }
break;
case 5:
{ return(LTtok); }
break;
case 6:
{ return(LEtok); }
break;
case 7:
{ return(EQtok); }
break;
case 8:
{ return(NEtok); }
break;
case 9:
{ return(GTtok); }
break;
case 10:
{ return(GEtok); }
break;
case 11:
{ return(install_id()); }
break;
case 12:
{ return(install_num()); }
break;
case 13:
{ return(ERRORtok); }
break;
case -1:
break;
default:
fprintf(yyout,"bad switch yylook %d",nstr);
} return(0); }
/* end of yylex */
        /* The other code to be included in lex.yy.c            */


/*************************************************************************/
/*                                                                       */
/* FUNCTION : install_id()                                               */
/* PURPOSE  : install_id() places the current lexeme into the           */
/*            identifier table                                           */
/*                                                                       */
/*************************************************************************/

install_id(){
        return(IDtok);
}

/*************************************************************************/
/*                                                                       */
/* FUNCTION : install_num()                                              */
/* PURPOSE  : install_num() takes a numeric token and installs it into  */
/*            a lexeme table of numbers.                                 */
/*                                                                       */
/*************************************************************************/

install_num(){
        return(NUMtok);
}
int yyvstop[] ={
0,

13,
0,

1,
13,
0,

1,
0,

12,
13,
0,
```

```
5,
13,
0,

7,
13,
0,

9,
13,
0,

11,
13,
0,

11,
13,
0,

11,
13,
0,

11,
13,
0,

12,
0,

6,
0,

8,
0,

10,
0,

11,
0,

11,
0,

2,
11,
0,

11,
0,

11,
0,

11,
0,

4,
11,
0,
```

```
3,
11,
0,
0};
# define YYTYPE unsigned char
struct yywork { YYTYPE verify, advance; } yycrank[] ={
0,0,    0,0,    1,3,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    1,4,    1,5,
4,5,    4,5,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    4,5,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    1,6,    6,14,   6,14,
6,14,   6,14,   6,14,   6,14,
6,14,   6,14,   6,14,   6,14,
0,0,    1,7,    1,8,    1,9,
7,15,   7,16,   1,10,   2,7,
2,8,    2,9,    9,17,   0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    0,0,    1,11,   12,20,
21,23,  13,21,  1,12,   22,24,
2,11,   11,19,  0,0,    23,25,
2,12,   0,0,    0,0,    0,0,
19,22,  1,13,   0,0,    0,0,
0,0,    0,0,    0,0,    2,13,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  0,0,    0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  0,0,
0,0,    0,0,    0,0,    0,0,
0,0,    10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  10,18,
10,18,  10,18,  10,18,  0,0,
0,0};
struct yysvf yysvec[] ={
0,      0,      0,
yycrank+-1,     0,              0,
yycrank+-7,     yysvec+1,       0,
yycrank+0,      0,              yyvstop+1,
yycrank+3,      0,              yyvstop+3,
yycrank+0,      yysvec+4,       yyvstop+6,
```

```
yycrank+2,      0,              yyvstop+8,
yycrank+3,      0,              yyvstop+11,
yycrank+0,      0,              yyvstop+14,
yycrank+9,      0,              yyvstop+17,
yycrank+76,     0,              yyvstop+20,
yycrank+1,      yysvec+10,      yyvstop+23,
yycrank+1,      yysvec+10,      yyvstop+26,
yycrank+1,      yysvec+10,      yyvstop+29,
yycrank+0,      yysvec+6,       yyvstop+32,
yycrank+0,      0,              yyvstop+34,
yycrank+0,      0,              yyvstop+36,
yycrank+0,      0,              yyvstop+38,
yycrank+0,      yysvec+10,      yyvstop+40,
yycrank+1,      yysvec+10,      yyvstop+42,
yycrank+0,      yysvec+10,      yyvstop+44,
yycrank+3,      yysvec+10,      yyvstop+47,
yycrank+6,      yysvec+10,      yyvstop+49,
yycrank+1,      yysvec+10,      yyvstop+51,
yycrank+0,      yysvec+10,      yyvstop+53,
yycrank+0,      yysvec+10,      yyvstop+56,
0,      0,      0};
struct yywork *yytop = yycrank+198;
struct yysvf *yybgin = yysvec+1;
char yymatch[] ={
00  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,011 ,012 ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
011 ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
'0' ,'0' ,'0' ,'0' ,'0' ,'0' ,'0' ,'0' ,
'0' ,'0' ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,
'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,
'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,
'A' ,'A' ,'A' ,01  ,01  ,01  ,01  ,01  ,
01  ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,
'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,
'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,'A' ,
'A' ,'A' ,'A' ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
01  ,01  ,01  ,01  ,01  ,01  ,01  ,01  ,
0};
char yyextra[] ={
0,0,0,0,0,0,0,0,
0,0,0,0,0,0,0,0,
0};
/*      ncform  4.1     83/08/11        */

int yylineno =1;
# define YYU(x) x
# define NLSTATE yyprevious=YYNEWLINE
char yytext[YYLMAX];
struct yysvf *yylstate [YYLMAX], **yylsp, **yyolsp;
char yysbuf[YYLMAX];
char *yysptr = yysbuf;
int *yyfnd;
extern struct yysvf *yyestate;
int yyprevious = YYNEWLINE;
yylook(){
        register struct yysvf *yystate, **lsp;
        register struct yywork *yyt;
        struct yysvf *yyz;
        int yych;
        struct yywork *yyr;
# ifdef LEXDEBUG
        int debug;
# endif
        char *yylastch;
        /* start off machines */
# ifdef LEXDEBUG
        debug = 0;
# endif
        if (!yymorfg)
                yylastch = yytext;
        else {
                yymorfg=0;
                yylastch = yytext+yyleng;
                }
        for(;;){
                lsp = yylstate;
                yyestate = yystate = yybgin;
                if (yyprevious==YYNEWLINE) yystate++;
                for (;;){
# ifdef LEXDEBUG
                        if(debug)fprintf(yyout,"state %d\n",yystate-yysvec-1);
# endif
                        yyt = yystate->yystoff;
                        if(yyt == yycrank){             /* may not be any transitions */
                                yyz = yystate->yyother;
                                if(yyz == 0)break;
                                if(yyz->yystoff == yycrank)break;
                                }
                        *yylastch++ = yych = input();
                tryagain:
# ifdef LEXDEBUG
                        if(debug){
                                fprintf(yyout,"unsigned char ");
                                allprint(yych);
                                putchar('\n');
                                }
# endif
                        yyr = yyt;
                        if ( (int)yyt > (int)yycrank){
                                yyt = yyr + yych;
                                if (yyt <= yytop && yyt->verify+yysvec == yystate){
                                        if(yyt->advance+yysvec == YYLERR)       /* error
transitions */
                                                {unput(*--yylastch);break;}
                                        *lsp++ = yystate = yyt->advance+yysvec;
                                        goto contin;
                                        }
                                }
# ifdef YYOPTIM
```

```
			else if((int)yyt < (int)yycrank) {		/* r < yycrank *
/
				yyt = yyr = yycrank+(yycrank-yyt);
# ifdef LEXDEBUG
				if(debug)fprintf(yyout,"compressed state\n");
# endif
				yyt = yyt + yych;
				if(yyt <= yytop && yyt->verify+yysvec == yystate){
					if(yyt->advance+yysvec == YYLERR)	/* error
 transitions */
						{unput(*--yylastch);break;}
					*lsp++ = yystate = yyt->advance+yysvec;
					goto contin;
					}
				yyt = yyr + YYU(yymatch[yych]);
# ifdef LEXDEBUG
				if(debug){
					fprintf(yyout,"try fall back character ");
					allprint(YYU(yymatch[yych]));
					putchar('\n');
					}
# endif
				if(yyt <= yytop && yyt->verify+yysvec == yystate){
					if(yyt->advance+yysvec == YYLERR)	/* error
 transition */
						{unput(*--yylastch);break;}
					*lsp++ = yystate = yyt->advance+yysvec;
					goto contin;
					}
				}
			if ((yystate = yystate->yyother) && (yyt= yystate->yystoff) != y
ycrank){
# ifdef LEXDEBUG
				if(debug)fprintf(yyout,"fall back to state %d\n",yystate
-yysvec-1);
# endif
				goto tryagain;
				}
# endif
			else
				{unput(*--yylastch);break;}
		contin:
# ifdef LEXDEBUG
			if(debug){
				fprintf(yyout,"state %d char ",yystate-yysvec-1);
				allprint(yych);
				putchar('\n');
				}
# endif
			;
			}
# ifdef LEXDEBUG
		if(debug){
			fprintf(yyout,"stopped at %d with ",*(lsp-1)-yysvec-1);
			allprint(yych);
			putchar('\n');
			}
# endif
		while (lsp-- > yylstate){
			*yylastch-- = 0;
			if (*lsp != 0 && (yyfnd= (*lsp)->yystops) && *yyfnd > 0){
				yyolsp = lsp;
				if(yyextra[*yyfnd]){		/* must backup */
					while(yyback((*lsp)->yystops,-*yyfnd) != 1 && ls
p > yylstate){
						lsp--;
						unput(*yylastch--);
						}
					}
				yyprevious = YYU(*yylastch);
				yylsp = lsp;
				yyleng = yylastch-yytext+1;
				yytext[yyleng] = 0;
# ifdef LEXDEBUG
				if(debug){
					fprintf(yyout,"\nmatch ");
					sprint(yytext);
					fprintf(yyout," action %d\n",*yyfnd);
					}
# endif
				return(*yyfnd++);
				}
			unput(*yylastch);
			}
		if (yytext[0] == 0  /* && feof(yyin) */)
			{
			yysptr=yysbuf;
			return(0);
			}
		yyprevious = yytext[0] = input();
		if (yyprevious>0)
			output(yyprevious);
		yylastch=yytext;
# ifdef LEXDEBUG
		if(debug)putchar('\n');
# endif
		}
	}
yyback(p, m)
	int *p;
{
if (p==0) return(0);
while (*p)
	{
	if (*p++ == m)
		return(1);
	}
return(0);
}
	/* the following are only used in the lex library */
yyinput(){
	return(input());
	}
yyoutput(c)
  int c; {
	output(c);
	}
yyunput(c)
   int c; {
	unput(c);
	}
```