```
# line 23 "trans.y"
#include "stdio.h"
#include <ctype.h>


# line 28 "trans.y"
typedef union  {
        int ival;
} YYSTYPE;
# define LPARENtok 257
# define RPARENtok 258
# define PLUStok 259
# define Ftok 260
# define Vtok 261
# define ERRORtok 262
#define yyclearin yychar = -1
#define yyerrok yyerrflag = 0
extern int yychar;
extern short yyerrflag;
#ifndef YYMAXDEPTH
#define YYMAXDEPTH 150
#endif
YYSTYPE yylval, yyval;
# define YYERRCODE 256

# line 56 "trans.y"


short yyexca[] ={
-1, 1,
        0, -1,
        -2, 0,
        };
# define YYNPROD 7
# define YYLAST 13
short yyact[]={

   4,   3,   9,   7,  10,   5,   1,   6,   2,   0,
   0,   0,   8 };
short yypact[]={

-260,-1000,-252,-256,-1000,-260,-1000,-259,-254,-1000,
-1000 };
short yypgo[]={

   0,   6,   8,   7 };
short yyr1[]={

   0,   1,   1,   2,   2,   3,   3 };
short yyr2[]={

   0,   4,   2,   1,   0,   2,   0 };
short yychk[]={

-1000,  -1,  -2, 261, 260, 257,  -3, 259,  -1, 261,
 258 };
short yydef[]={

   4,  -2,   0,   6,   3,   4,   2,   0,   0,   5,
   1 };
#ifndef lint
static char yaccpar_sccsid[] = "@(#)yaccpar     4.1     (Berkeley)      2/11/83";
#endif
```

```
#
# define YYFLAG -1000
# define YYERROR goto yyerrlab
# define YYACCEPT return(0)
# define YYABORT return(1)

/*      parser for yacc output  */

#ifdef YYDEBUG
int yydebug = 0; /* 1 for debugging */
#endif
YYSTYPE yyv[YYMAXDEPTH]; /* where the values are stored */
int yychar = -1; /* current input token number */
int yynerrs = 0;  /* number of errors */
short yyerrflag = 0;  /* error recovery flag */

yyparse() {

        short yys[YYMAXDEPTH];
        short yyj, yym;
        register YYSTYPE *yypvt;
        register short yystate, *yyps, yyn;
        register YYSTYPE *yypv;
        register short *yyxi;

        yystate = 0;
        yychar = -1;
        yynerrs = 0;
        yyerrflag = 0;
        yyps= &yys[-1];
        yypv= &yyv[-1];

 yystack:    /* put a state and value onto the stack */

#ifdef YYDEBUG
        if( yydebug  ) printf( "state %d, char 0%o\n", yystate, yychar );
#endif
                if( ++yyps> &yys[YYMAXDEPTH] ) { yyerror( "yacc stack overflow" ); r
eturn(1); }
                *yyps = yystate;
                ++yypv;
                *yypv = yyval;

 yynewstate:

        yyn = yypact[yystate];

        if( yyn<= YYFLAG ) goto yydefault; /* simple state */

        if( yychar<0 ) if( (yychar=yylex())<0 ) yychar=0;
        if( (yyn += yychar)<0 || yyn >= YYLAST ) goto yydefault;

        if( yychk[ yyn=yyact[ yyn ] ] == yychar ){ /* valid shift */
                yychar = -1;
                yyval = yylval;
                yystate = yyn;
                if( yyerrflag > 0 ) --yyerrflag;
                goto yystack;
                }

 yydefault:
        /* default state action */

        if( (yyn=yydef[yystate]) == -2 ) {
                if( yychar<0 ) if( (yychar=yylex())<0 ) yychar = 0;
```

```
                /* look through exception table */

                for( yyxi=yyexca; (*yyxi!= (-1)) || (yyxi[1]!=yystate) ; yyxi += 2 )
 ; /* VOID */

                while( *(yyxi+=2) >= 0 ){
                        if( *yyxi == yychar ) break;
                        }
                if( (yyn = yyxi[1]) < 0 ) return(0);   /* accept */
                }

        if( yyn == 0 ){ /* error */
                /* error ... attempt to resume parsing */

                switch( yyerrflag ){

                case 0:   /* brand new error */

                        yyerror( "syntax error" );
                yyerrlab:
                        ++yynerrs;

                case 1:
                case 2: /* incompletely recovered error ... try again */

                        yyerrflag = 3;

                        /* find a state where "error" is a legal shift action */

                        while ( yyps >= yys ) {
                           yyn = yypact[*yyps] + YYERRCODE;
                           if( yyn>= 0 && yyn < YYLAST && yychk[yyact[yyn]] == YYERR
CODE ){

                              yystate = yyact[yyn];  /* simulate a shift of "error"
*/

                              goto yystack;
                              }
                           yyn = yypact[*yyps];

                           /* the current yyps has no shift onn "error", pop stack *
/

#ifdef YYDEBUG
                           if( yydebug ) printf( "error recovery pops state %d, unco
vers %d\n", *yyps, yyps[-1] );
#endif
                           --yyps;
                           --yypv;
                           }

                        /* there is no state on the stack with an error shift ... ab
ort */

        yyabort:
                        return(1);


                case 3:  /* no shift yet; clobber input char */

#ifdef YYDEBUG
                        if( yydebug ) printf( "error recovery discards char %d\n", y
ychar );
#endif


                        if( yychar == 0 ) goto yyabort; /* don't discard EOF, quit *
/
                        yychar = -1;
                        goto yynewstate;   /* try again in the same state */

                        }

                }

        /* reduction by production yyn */

#ifdef YYDEBUG
                if( yydebug ) printf("reduce %d\n",yyn);
#endif
                yyps -= yyr2[yyn];
                yypvt = yypv;
                yypv -= yyr2[yyn];
                yyval = yypv[1];
                yym=yyn;
                        /* consult goto table to find next state */
                yyn = yyr1[yyn];
                yyj = yypgo[yyn] + *yyps + 1;
                if( yyj>=YYLAST || yychk[ yystate = yyact[yyj] ] != -yyn ) yystate =
 yyact[yypgo[yyn]];
                switch(yym){

case 1:
# line 41 "trans.y"
{printf("   E --> Prefix(E)\n"); } break;
case 2:
# line 43 "trans.y"
{printf("   E --> V Tail\n"); } break;
case 3:
# line 46 "trans.y"
{printf("   Prefix --> F\n"); } break;
case 4:
# line 48 "trans.y"
{printf("   Prefix --> (empty)\n"); } break;
case 5:
# line 51 "trans.y"
{printf("   Tail --> +V\n"); } break;
case 6:
# line 53 "trans.y"
{printf("   Tail --> (empty)\n"); } break;
                }
                goto yystack;  /* stack new state and value */

        }
```