Chapter 4

```
enum token {
    INT,
    ID,
    ASSIGN,
    PRINT,
    LPAREN,
    RPAREN,
    PLUS,
    MINUS,
    TIMES,
    DIV,
    SEMICOLON,
    COMMA
};
union tokenval {
    int num;
    string id;
};
enum token tok;
union tokenval tokval;
void advance() { tok = getToken(); }
void eat(enum token t) {
    if (tok == t)
        advance();
    else
        error();
}
typedef struct table *Table_;
Table_ {
    string id;
    int value;
    Table_tail
};
Table_ Table(string id, int value, struct table *tail);
Table_ table = NULL;
int lookup(Table_ table, string id) {
    assert(table != NULL);
    if (id == table.id)
        return table.value;
    else
        return lookup(table.tail, id);
}
void update(Table_ *tabptr, string id, int value) {
    *tabptr = Table(id, value, *tabptr);
}
void eatOrSkipTo(int expected, int *stop) {
    if (tok == expected)
        eat(expected);
    else
```

```
        skipto(stop);
}
int P_follow = [];
int P(void) {
    switch (tok) {
        case ID:
        case PRINT:
            return S();
        default:
            printf("expect ID or PRINT!\n");
            return 0;
    }
}
int S_follow = [SEMICOLON, COMMA] ;
int S(void) {
    switch (tok) {
        case ID:
            string id = tokval.id;
            eat(ID);
            eat(ASSIGN);
            update(table, id, E());
            return T();
        case PRINT:
            eat(PRINT);
            eat(LPAREN);
            L();
            printf("\n");
            eatOrSkipTo(RPAREN, S_follow);
            return T();
        default:
            printf("expect ID or PRINT!\n");
            skipto(S_follow);
            return 0;
    }
}
int T_follow = [SEMICOLON, COMMA];
void T(void) {
    switch (tok) {
        case SEMICOLON:
            eat(SEMICOLON);
            S();
            T();
            return 0;
        case COMMA:
            skipto(T_follow);
            return 0;
        default:
            printf("expect SEMICOLON or COMMA!\n");
            skipto(T_follow);
            return 0;
    }
}
int L_follow = [RPAREN] ;
void L(void) {
    switch (tok) {
        case ID:
        case INT:
        case PRINT:
```

```
        case LPAREN:
            printf("%d ", E());
            R();
            return 0;
        default:
            printf("expect ID, INT, PRINT, or LPAREN!\n");
            skipto(L_follow);
            return 0;
    }
}
int R_follow = [RPAREN] ;
void R(void) {
    switch (tok) {
        case COMMA:
            eat(COMMA);
            printf("%d ", E());
            R();
            return 0;
        case RPAREN:
            skipto(R_follow);
            return 0;
        default:
            skipto(R_follow);
            printf("expect COMMA or RPAREN!\n");
    }
}
int E_follow = [ SEMICOLON, COMMA, RPAREN ];
int E(void) {
    switch (tok) {
        case ID:
            if (lookahead() == ASSIGN) {
                S();
                eat(COMMA);
                return E();
            } else
                return M(P());
        case INT:
            return M(P());
        case PRINT:
            S();
            eat(COMMA);
            return E();
        case LPAREN:
            return M(P());
        default:
            printf("Expect ID, INT, PRINT, or LPAREN!\n");
            skipto(E_follow);
            return 0;
    }
}
int P_follow = [ SEMICOLON, COMMA, RPAREN, PLUS, MINUS ];
void P(void) {
    switch (tok) {
        case ID:
        case INT:
        case LPAREN:
            return Q(F());
        default:
```

```
            printf("expect ID, INT or LPAREN!\n");
            skipto(P_follow);
            return 0;
    }
}
int M_follow = [ SEMICOLON, COMMA, RPAREN ];
int M(int x) {
    switch (tok) {
        case PLUS:
            eat(PLUS);
        return M(x + P()) case MINUS:
            eat(MINUS);
            return M(x - P());
        default:
            skipto(M_follow);
            return x;
    }
}
int F_follow = [ SEMICOLON, COMMA, RPAREN, PLUS, MINUS, TIMES, DIV ];
int F(void) {
    switch (tok) {
        case ID:
            return lookup(table, tokval.id);
        case INT:
            return tokval.num;
        case LPAREN:
            eat(LPAREN);
            int res = E();
            eatOrSkipTo(RPAREN, Q_follow);
            return res;
        default:
            printf("Expect ID, INT or LPAREN!\n");
            skipto(Q_follow);
            return 0;
    }
}
int Q_follow = [ SEMICOLON, COMMA, RPAREN, PLUS, MINUS ];
int Q(int x) {
    switch (tok) {
        case TIMES:
            eat(TIMES);
            return Q(x * F());
        case DIV:
            eat(DIV);
            return Q(x / F());
        default:
            skipto(Q_follow);
            return x;
    }
}
```