HW 3

```
enum token {ID, INT, PLUS, MINUS, TIMES, DIV, ASSIGN, PRINT, LPAREN, RPAREN,
SEMICOLON, COMMA};
union tokenval {string id; int num;};
enum token tok;
union tokenval tokval;
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);
}

int S_FOLLOW[] = {SEMICOLON, COMMA};
void S (void) {
    switch(tok) {
        case ID:{
                    string id=tokval.id;
                    if(lookahead()==ASSIGN) {
                        advance();
                        update(table,id,E());
                    }
                    skipto(S_FOLLOW);
                    break;
                }
        case PRINT:{
                        advance();
                        if(lookahead()==LPAREN) {
                            L();
                            eatOrSkipTo(RPAREN,S_FOLLOW);
                        }
                        break;
                    }
        default:{
                    printf("expected ID or PRINT");
                    skipto(S_FOLLOW);
                }
    }

int L_FOLLOW[] = {COMMA, SEMICOLON, RPAREN};
void L (void) {
    switch(tok) {
        case ID:
        case INT:{
                        printf("%d",E());
                        break;
                }
        default:{
                        printf("expected ID or INT");
                        skipto(L_FOLLOW);
                }
    }
}

int E_FOLLOW[] = {SEMICOLON, PLUS, MINUS, TIMES, DIV, RPAREN};
int E() {
    switch(tok) {
        case ID:{
                    int i = lookup(table,tokval.id);
                    int p = lookahead();
                    if ( p == PLUS || P == MINUS || P == TIMES || P == DIV ) {
                        advance();
                        return B(i);
                    }
                    advance();
                    return i;
                }
        case INT:{
                    int i = tokval.num;
                    int p = lookahead();
                    if ( p == PLUS || P == MINUS || P == TIMES || P == DIV ) {
                        advance();
                        return B(i);
                    }
                    advance();
                    return i;
                }
        case LPAREN:{
                        int i = E();
                        if ( lookahead() == RPAREN )
                            return i;
                        else{
                            skipto(E_FOLLOW);
                            return 0;
                        }
                    }
        default:{
                    printf("expected ID,INT,LPAREN");
                    skipto(E_FOLLOW);
                    return 0;
                }
    }
}

int B_FOLLOW[] = {ID, INT, LPAREN};
int B(int a) {
    switch(tok) {
        case PLUS: {
                        advance();
                        return (a + E());
                    }
        case MINUS: {
                        advance();
```

```
                                return (a - E());
                            }
            case TIMES: {
                                advance();
                                return (a * E());
                            }
            case DIV: {
                                advance();
                                return (a / E());
                            }
            default: {
                                Skipto(B_FOLLOW);
                                return 0;
                            }
    }
}

void eatOrSkipTo(int expected,int *stop)
{
    if ( tok == expected ) {
        eat(expected);
    }
    else {
        skipto(stop);
    }
}
```