```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "lexical.h"
#include "nextInputChar.h"
#include "tokenStack.h"

/*
 *This is my solution to the lab5 of eecs2031 about builting a RPN Calculator.
 *Name: Yuxian Wang
 *Student Number: 215170418
 *Date: 2017/11/03
 */

/*pop the top element off of the stack */
static int popInt(struct tokenStack *s) {
    if (s->top < 0) {
        printf("popInt: error, aborting.\n");
        exit(0);
    }
    struct lexToken *token;
    token = allocToken();
    token = popTokenStack(s);
    freeToken(token);
    return atoi(token->symbol);
}

/* take an int and create a lexToken */
static void pushInt(struct tokenStack *s, int v) {
    struct lexToken *token;
    token = allocToken();
    token->kind = LEX_TOKEN_NUMBER; /* holds a LEX_TOKEN_NUMBER */
    sprintf(token->symbol, "%c", v); /* push it on the stack*/
    pushTokenStack(s, token);
}

static void doOperator(struct tokenStack *s, char *op) {
    if (!strcmp(op, "quit")) {
        exit(0);
    } else if (!strcmp(op, "print")) {
        struct lexToken *t = popTokenStack(s);
        dumpToken(stdout, t);
        freeToken(t);
    } else {
        fprintf(stderr, "unknown command %s\n", op);
        exit(1);
    }
}

int main(int argc, char *argv[]) {
    setFile(stdin);
    struct tokenStack *stack;
    stack = createTokenStack();
    struct lexToken *token;
    int kind, num1, num2;
    char op;
    while (token = nextToken()) {
        kind = token->kind;
        switch (kind) {
        case LEX_TOKEN_NUMBER:
            pushInt(stack, atoi(token->symbol));
            break;
        case LEX_TOKEN_IDENTIFIER:
            doOperator(stack, token->symbol);
            break;
        case LEX_TOKEN_OPERATOR:
            switch (token->symbol[0]) {
            case '+':
                pushInt(stack, popInt(stack) + popInt(stack));
                break;
            case '-':
                num1 = popInt(stack);
                num2 = popInt(stack);
                pushInt(stack, num1 - num2);
                break;
            case '*':
                pushInt(stack, popInt(stack) * popInt(stack));
                break;
            case '/':
                num1 = popInt(stack);
                num2 = popInt(stack);
```

```c
                pushInt(stack, num2 / num1);
                break;
            default:
                printf("wrong\n");
            }
            break;
        case LEX_TOKEN_EOF:
            printf("end of line, quit.\n");
            exit(1);
            break;
        default:
            printf("wrong");
        }
    }
    return 0;
}

```