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

static int popInt(struct tokenStack *s)
{
  /* write this */
  /* If this cannot be done, your program should print an error message end quit */
      if (s->top <= 0){
      fprintf(stderr , " popInt : popping an empty stack,aborting \n ");
      exit(1);
      }

  /*popInt should take the stack(*s) and pop the top element off of the stack  */
      struct lexToken *topToken= popTokenStack(s);
      char number = topToken->symbol[0];

  /*returning its int value*/
      int  intnum = number - '0';
      /*free(topToken);*/
      return intnum;
}

static void pushInt(struct tokenStack *s, int v)
{
  /* write this */

 /* pushInt should take an int(v)*/
 /* create a lexToken that holds a LEX_TOKEN_NUMBER and push that on the stack*/
   struct lexToken *newtoken = allocToken();
   newtoken->kind = LEX_TOKEN_NUMBER;

   char numchar = v +'0';
   /*char syb[]={numchar,'\0'};*/
   newtoken->symbol[0] = numchar;
   newtoken->symbol[1] = '\0'; /* ? int to char */
   pushTokenStack(s,newtoken);
   /*free(newtoken);*/
}

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,"don't know |%s|\n", op);
    exit(1);
  }
}

int main(int argc, char *argv[])
{
  setFile(stdin);
  struct tokenStack *stack = createTokenStack();

  /* write this */
```

```
  /* Your code should continue to read tokens from the input [*nexttoken()]  */

  /* For tokens of type LEX_TOKEN_EOF your code should quit
     For tokens of type LEX_TOKEN_IDENTIFIER your code should call doOperator
     For tokens of type LEX_TOKEN_NUMBER your code should push the token on the stack *
/
     int type = 0;
     do {
     struct lexToken *next = nextToken();
     int type = next->kind;
     char input = next-> symbol[0];
     char *identString= next->symbol;
     double op2;

     switch (type){
     /*case LEX_TOKEN_EOF :
          break;
     */
     case LEX_TOKEN_IDENTIFIER :
         doOperator(stack,identString);
         break;
     case LEX_TOKEN_NUMBER:
         pushTokenStack(stack, next);
         break;
     case LEX_TOKEN_OPERATOR:
         switch (input){
         case '+':
         pushInt(stack , popInt(stack) +        popInt(stack) );
         break;
         case '*':
         pushInt(stack , popInt(stack) *        popInt(stack) );
         break;
         case '-':
         op2 = popInt(stack);
         pushInt(stack , popInt(stack) - op2     );
         break;
         case '/':
         op2 = popInt(stack);
         if (op2 != 0.0)
         pushInt(stack , popInt(stack) / op2     );
         else
         fprintf(stderr , " zero divisor \n ");
         break;
         }
     }
     }
     while (type != LEX_TOKEN_EOF ); /* not sure what */


     fprintf(stdout,"%d \n", popInt(stack));

  /* For tokens of type LEX_TOKEN_OPERATOR your code should pop the top two elements of
f of the stack,
     Perform the corresponding integer operation of the top two elements and
     push the corresponding result on the stack.*/

}
```