```
/*-------------------------------------------------------------------------+
File Name: readDate.c
+-------------------------------------------------------------------------+

+-------------------------------------------------------------------------+
   Programming III COP4338
   Author: Daniel Gonzalez P#4926400
   assignment 5: Date Validate / Format
   Date: 11/08/2016 ELLECTION DAY!!!!

  program description
     Input: Accept input for the first program via the command-line arguments.
            Input will be the number of valid entries to be redirected from
            the dates input file (dates.dat). A zero indicates to input all
            entries from the dates input file. Your program should validate
            (day, month & year - see page 111 for validation ideas) and skip
            corrupt dates in the dates.dat file (see page 159 for scanning
            ideas). This validated input will then be piped out to the
            second program.
            The second program will accept these validated dates in the
            month/day/year format and convert each of them to the day,
            abbreviated month & year format – both exhibited above. The
            abbreviated month should consist of the first three letters of
            the month, capitalized. These converted results will be redirected
            to the output file (output.dat), followed by a copy of the
            original (dates.dat) data.

     Output: Generates an output file (output.dat) that contains a
             converted list of dates in day, abbreviated month & year
             format (i.e. 1 JAN 1900), followed by the original list of
             dates in month/day/year format (i.e. 1/1/1900). This output file
             will be the result of appending the input file (dates.dat), which
             is accessed by the first program, with the result output file
             (output.dat), generated by the second program.


  +--------------------------------------------------------------------------+
  | I Daniel Gonzalez #4926400 hereby certify that this collective work is   |
  | my own and none of it is the work of any other person or entity.         |
  +--------------------------------------------------------------------------+


  how to compile and execute:
    1.Open the terminal
        Go to the program folder that contains all the files required for
        the program to compile including all header files(*.h).
        Run the following command "make"

    2.Open the terminal
        Go to the program folder that contains all the files required for
        the program to compile including all header files(*.h).
        COMPILE: "gcc -Wall -w -lm readDate.c dateValidate.c -o validateDate"

    Program execution:
    From the terminal enter:
     "./validateDate < dates.dat [X] | ./format > output.dat"

     X: is the amount of validated dates




+-------------------------------------------------------------------------*/

#include "general.h"



/**-------------------------------------------------------------------------+
```

```
    //Get a line of data.
    //Validate the line.
        //Print data with correct format.
        //Write out the formatted data.
        //If error enabled then write the error.
    //Write out the original data.
 */
int main(int argc, char *argv[])
{

    /* Declare variables  */
    LineList inputLines;
    DateKey date;
    Data data;
    Boolean processAll = FALSE;

    /* Initialize variables */
    initializeLineList(&inputLines);

    /* Input values */
    int maxLines = (argc == MAX_AMOUNT_ALLOWED_ARGS) ? atoi(argv[1]): 0;
    if (maxLines == 0){
        processAll = TRUE;
    }else{
    }

    /* Main process */
    while ((fgets(data.input, MAX_LINE, stdin) != NULL) &&
            (maxLines > 0 || processAll)) {

        date = getDateKeys(data.input);
        if(date.error == NO_ERRORS){
            printfDate(data.output, DATE_FORMAT_SLASH, &date);
            fputs(data.output, stdout);

            maxLines--;

        }else{
            if(ENABLE_ERROR){
                printError(data.error, data.input, date.error);
            }else{
            }
        }

        appendToLineList(&inputLines, data.input);
    }

    fputc(HAND_SHAKING_SIGNAL, stdout);

    fprintf(stdout, "\n\n%s\n", LB_ORIGINAL_DATA);
    int i = 0;
    for(i = 0; i < inputLines.size; i++){
        fputs(inputLines.lines[i].content,stdout);
    }

    freeLineList(&inputLines);

    /* Output results */
    return NO_ERRORS;
}
```