I have used "more" notation to include several files in one file;
each file name is preceeded and followed by a line with colons (:::::)
and each files contents is below its colon–surrounded name.

I have written this as a collection of C++ .h files, but with private and
protected data missing, each one for one module
of Parnas's second decomposition. It is assumed that all and only the
functions listed herein are public and all others that might show up in the
implementation are private.

Anything that is private will be declared ONLY in the .c files.

Note that I have made the procedure and function names more descriptive of
what they do. I have a choice of very descriptive names OR briefer, less
descriptive names with more comments. I prefer the former.

```
::::::::::::::::::::::::::::
line_storage.h
::::::::::::::::::::::::::::
class LineStorage {

/* class of object providing storage of all of the lines to be indexed */

protected;

    const int MAXCHARS = ?????;
    const int MAXLINES = ?????;
    char the_words[MAXCHARS]
    stuct pair {
                int line_no;
                int char_no;
    };
    pair line_index[MAXLINES];

public:

    LineStorage() {...};
    ~LineStorage() {...};

    void set_a_char_in_a_word_of_a_line(int line_index, int word_index,
                int char_index, char c) {...};

    /* for all l, w, and c, get_a_char_in_a_word_of_a_line(l,w,c) is defined
                only after doing set_a_char_in_a_word_of_a_line(l,w,c,d) for some
                d */

    char get_a_char_in_a_word_of_a_line(int line_index, int word_index,
                int char_index) {...};

    /* the following are defined ONLY after set_a_char_in_a_word_of_a_line has
                been executed to fill the line_storage */
    int no_of_chars_in_a_word_of_a_line (int line_index, int word_index) {...};
    int no_of_words_in_a_line (int line_index) {...};
    int no_of_lines() {...};

};
```

```
::::::::::::::::::::::::::::
circular_shifts.h
::::::::::::::::::::::::::::
#include line_storage.h
class CircularShifts: private LineStorage {

/* class of object providing storage of all of the circular shifts of the
   lines to be indexed */

protected;

   pair cs_line_index[5*MAXLINES];

public:

   CircularShifts() {...};
   ~CircularShifts() {...};

   void set_up_circular_shifts_of_lines_from_line_storage(LineStorage ls);

   /* the following are defined ONLY after
                  set_up_circular_shifts_of_lines_from_line_storage
                  has been executed to fill the circular_shifts */

   char get_a_char_in_a_word_of_a_cs_line(int cs_line_index,
                  int word_index, int char_index) {...};
   int no_of_chars_in_a_word_of_a_cs_line (int cs_line_index,
                  int word_index) {...};
   int no_of_words_in_a_cs_line (int cs_line_index) {...};
   int no_of_cs_lines() {...};

};
```

```
::::::::::::::::::::::::::::
alphabetized_circular_shifts.h
::::::::::::::::::::::::::::
#include circular_shifts.h
class AlphabetizedCircularShifts {

/* class of object providing storage of alphabetized list of the circular
                shifts of the lines to be indexed */

protected;

    pair alphed_cs_line_index[5*MAXLINES];


public:

    AlphabetizedCircularShifts() {...};
    ~AlphabetizedCircularShifts() {...};

    void create_alphabetized_circular_shifts_from_circular_shifts
                (CircularShifts& cs) {...};
    int index_in_circular_shifts_of_ith_in_alphabetical_ordering(int i) {...};

};
```

```
::::::::::::::::::::::::::::
input.h
::::::::::::::::::::::::::::
#include line_storage.h
#include <stdio.h>
void input_lines_into_line_storage(LineStorage& ls) {...};

::::::::::::::::::::::::::::
output.h
::::::::::::::::::::::::::::
#include alphabetized_circular_shifts.h
#include <stdio.h>
void output_alphabetized_circular_shifts_in_sophisticated_way
    (AlphabetizedCircularShifts acs, LineStorage ls,CircularShifts cs) {...};
```

```
::::::::::::::::::::::::::::
kwic.c
::::::::::::::::::::::::::::
#include alphabetized_circular_shifts.h
#include input.h
#include output.h

LineStorage ls;
CircularShifts cs;
AlpabetizedCircularShifts acs;

void main(){
    input_lines_into_line_storage(&ls);
    acs.create_alphabetized_circular_shifts_from_circular_shifts(&cs);
    output_alphabetized_circular_shifts_in_sophisticated_way(acs, ls, cs);
}
```