```
package words is
        type word(size:integer);
        procedure create_word(w:in out word,s:string);
        procedure insert_char_into_word(w:in out word,c:character);
        procedure insert_string_into_word(w:in out word,s:string);
        function get_string_of_word(w:word) returns string;
        function get_char_of_word(w:word, i:integer) returns character;
        function length(w:word) returns integer;
        -- sorting alphabetically
        function < (w1,w2:word) returns boolean;
        function <= (w1,w2:word) returns boolean;
        function > (w1,w2:word) returns boolean;
        function >= (w1,w2:word) returns boolean;
        function = (w1,w2:word) returns boolean;
        function /= (w1,w2:word) returns boolean;
end word;
```

```
with word;
package lines is
        type line(size:integer); --size is number of words
        procedure create_line(l:in out line);
        procedure insert_word_into_line(l:in out line,w:word);
        procedure circular_shift_line(l:in out line);
        procedure make_circular_shift_line(l:in line, k:out line);
        function generate_circular_shift_line(l:line) returns line;
        function get_string_of_line(l:line) returns string;
        function get_word_of_line(l:line;i:integer) returns word;
        function get_char_of_line(l:line, i:integer) returns character;
        function get_char_of_word_of_line(l:line,c,w:integer)
                                        returns character;
        function no_words(l:line) returns integer;
        function no_chars(l:line) returns integer;
        -- sorting alphabetically
        function < (l1,l2:line) returns boolean;
        function <= (l1,l2:line) returns boolean;
        function > (l1,l2:line) returns boolean;
        function >= (l1,l2:line) returns boolean;
        function = (l1,l2:line) returns boolean;
        function /= (l1,l2:line) returns boolean;
end lines;
```

Method 1:

```
with words, lines
package line_storage is
                -- storage consists of array of new lines
                --
end line_storage;

with words, lines
package circular_shifts is
                -- storage consists of array of new lines
                --
end circular_shifts;

with words, lines
package alphed_circular_shifts is
                -- storage consists of array of new lines
                --
end alphed_circular_shifts;
```

Method 2:

```
with words, lines
package line_storage is
                -- storage consists of array of new lines
                --
end line_storage;

with words, lines
package circular_shifts is
                -- storage consists of array of new lines
                --
end circular_shifts;

with words, lines
package alphed_circular_shifts is
                -- storage consists of array of pointers to old lines
                --
end alphed_circular_shifts;
```

Not much good to have circular_shifts be pointers to old lines

Could conceivably implement these lines in terms of indexing tables for charlists and charlists