In homework #3, you will add two functions to enhance the capability of the text editor. One function is **Text editing mode**, and the other is **Query Facility**. For developing this homework#3, you can choose either the program you finished in homework#2 or the sample program provided by TA as your base program.

1. **Text editing modes (50%)**

The editor should have two editing modes: **character-based** and **word-based**. You have accomplished the **word-based** mode in homework#2. In homework#3, you must complete the other mode - **character-based**.

The following items describe the requirement features of **character-based** mode:

A. The default editing mode of the editor is **character-based mode**. (2%)

B. The **environment variable** of the editor is used to change the behavior of the editor; therefore, a command “set” is introduced to configure the environment variables. Users can use “set mode [argument]” command to change the editing mode (i.e. behavior) of the editor. The command “set mode char” is used to change the editing mode to character-based; the other command “set mode word” is used to change the editing mode to word-based. (3%)

C. Each operation for editing, beside the **map** command, would focus on dealing with characters (i.e., string) in character-based mode. Table 1 shows the further descriptions. (40%)

<table>
<thead>
<tr>
<th>Command</th>
<th>Arguments</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. goto</td>
<td>“x” “y”</td>
<td>The cursor directly is moved to the specify character-based location.</td>
</tr>
<tr>
<td>2. search</td>
<td>“string”</td>
<td>To search the string in the text.</td>
</tr>
<tr>
<td>3. insert</td>
<td>“string”</td>
<td>To insert the string into current location.</td>
</tr>
<tr>
<td></td>
<td>“string” “x” “y”</td>
<td>To insert the string into specific location.</td>
</tr>
<tr>
<td>4. delete</td>
<td>“x” “y”</td>
<td>To delete a string at current location.</td>
</tr>
<tr>
<td></td>
<td>“x” “y”</td>
<td>To delete a character at the specific location.</td>
</tr>
<tr>
<td>5. modify</td>
<td>“char”</td>
<td>To modify a character at current location.</td>
</tr>
<tr>
<td></td>
<td>“string” “x” “y”</td>
<td>To modify a character at the specific location.</td>
</tr>
<tr>
<td>6. replace</td>
<td>“string1” “string2”</td>
<td>To replace the string#1 to string#2. This operation must work through the text.</td>
</tr>
<tr>
<td>7. copy</td>
<td></td>
<td>To copy a character at current location.</td>
</tr>
<tr>
<td>8. paste</td>
<td></td>
<td>To paste a character (which had be copied) into current location.</td>
</tr>
</tbody>
</table>
D. The function of the `map` command doesn’t change in both modes, because the text location map must works based on higher semantics of words.

E. The value of the `cursor` of the editor also has two modes: word-based and character-based. That is, the moving unit of cursor is either by character or by word. Note that in character-based mode, the editor treats any kinds of character as minimum moving unit. Fig 1 shows the possible values of the cursor in character-based mode. (5%)

![Fig 1. The possible values of cursor in character-based mode](image)

2. **Query Facility (40%)**

The editor has only supported simple search, i.e. full word match. You will implement a **Query Facility** that can do more powerful query than the simple search. It could improve your understanding for Query Facility by studying chapter 17 (p949) of the textbook.

Of course, the different text editing modes of the editor affect the behavior of Query Facility. In character-based mode, Query Facility is focused on dealing with **string (compose of characters)**. In word-based mode, Query Facility is focus on dealing with **word** of which the meaning is already defined in previously homework.

The following items describe the requirement features for Query Facility:

A. A new command “query” is introduced for request the query services provided by Query Facility. Note that such query services is case insensitive. (3%)

B. **Fuzzy string/word match by using character “*”**. Note that this feature should work on both command “search” and “query”. (3%)

C. **A single string/word query**. All lines in which the word appears are displayed with the line number in parentheses. (The lines are displayed in ascending order). (3%)

```
(0, 0)>query daddy
Requested query: daddy
daddy (3) lines match
(1) Alice Emma has long flowing red hair. Her Daddy says
(4) magical but untamed. "Daddy, shush, there is no such thing,"
(6) Shyly, she asks, "I mean, Daddy, is there?"
```

D. **A Not Query, using the ! operator**. All lines in which the name does not appear within the text are displayed. (3%)
E. An **Or Query, using the **||**  operator. All lines in which either of the two names appear within the text are displayed. (3%)  

```
(0, 0)>query fiery || untamed
Requested query: fiery || untamed
fiery || untamed ( 2 ) lines match
(3) like a fiery bird in flight. A beautiful fiery bird, he tells her,
(4) magical but untamed. "Daddy, shush, there is no such thing,"
```

F. An **And Query, using the **&&**  operator. All lines in which both words are not only present but adjacent within the text are displayed. This includes the last word of a line and the first word of the subsequent line. (3%)  

```
(0, 0)>query fiery && bird || shyly
Requested query: fiery && bird || shyly
fiery && ( bird || shyly ) ( 1 ) lines match
(3) like a fiery bird in flight. A beautiful fiery bird, he tells her,
(6) Shyly, she asks, "I mean, Daddy, is there?"
```

G. **These operators can be combined. (7%)  

```
(0, 0)>query fiery && bird || shyly
Requested query: fiery && bird || shyly
fiery && bird || shyly ( 2 ) lines match
(3) like a fiery bird in flight. A beautiful fiery bird, he tells her,
(6) Shyly, she asks, "I mean, Daddy, is there?"
```

H. To allow **subgrouping of a query, Query Facility must also support parentheses ( and ). (15%)  

```
(0, 0)>query fiery && ( bird || shyly )
Requested query: fiery && ( bird || shyly )
fiery && ( bird || shyly ) ( 1 ) lines match
(3) like a fiery bird in flight. A beautiful fiery bird, he tells her,
```

3. **Unit Test (10%)**

You must continue to maintain your existing test cases. In addition, you need to design new test cases for the newly added functionalities. All the programs (classes and function) of the editor must pass the unit testing. Do your best.