Assignment I — Structs and Arrays

TA: Hank(s0963360045@gmail.com)

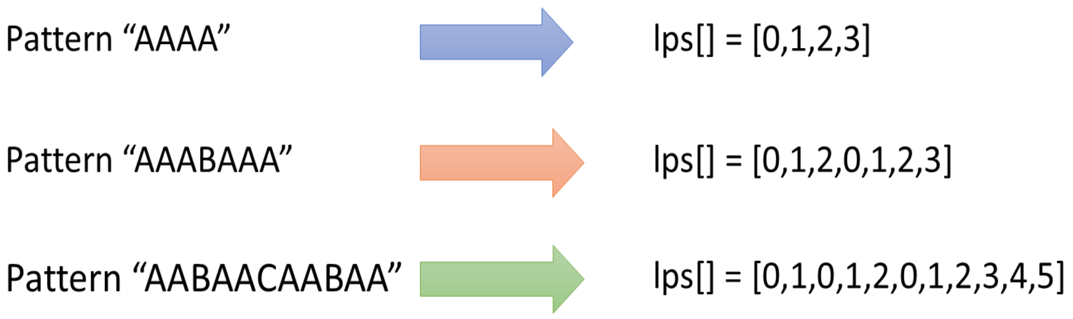
Deadline: 2019 Oct. 15, 11:59pm

In this homework, you will be asked to implement the ROT13 and KMP algorithms to find the secret message from a cipher text using structures and arrays.

First, you are required to implement the ROT13 algorithm to find the key by decrypting a given secret string. Next, after finding the key, utilize the KMP algorithm to search the matched words with the key and find all the positions of the matched words in the given article. The last step is to map a key index with a corresponding ASCII character and its phrase in the given dictionary to compose a secret message. The details of KMP, ROT13, and the implementation steps are described in the following sections.

- KMP

Pattern searching is an important problem in computer science. When we search for a string in a word file, a website, or a database, a pattern searching algorithm is used to compute the search results efficiently. The KMP algorithm improves the worst-case complexity to O(n). Given a string txt[], the KMP algorithm preprocesses a pattern pat[] and constructs an auxiliary lps[] (**the longest proper prefix/suffix array**) of size m (equal to the size of the pattern) which is used to skip the number of characters while matching the text. The details about lps[] can be referred to the KMP algorithm in the textbook. Figure 1 shows three patterns their values of lps[].



Pattern “AAAA” → lps[] = [0,1,2,3]

Pattern “AAABAAA” → lps[] = [0,1,2,0,1,2,3]

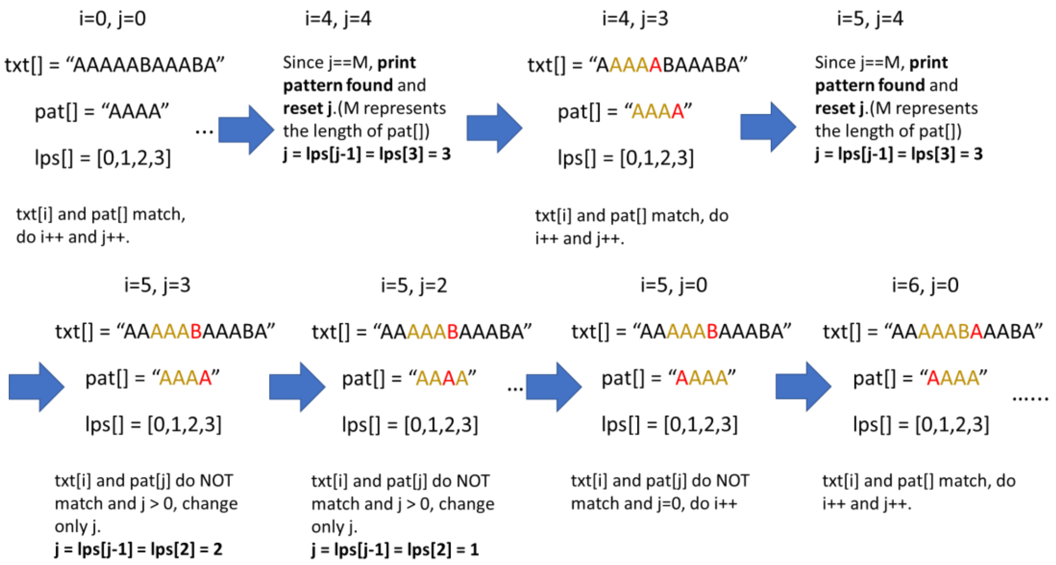
Pattern “AABAACAABAA” → lps[] = [0,1,0,1,2,0,1,2,3,4,5]

Figure 1: The examples of lps[].

lps[] is used to determine the next character to be matched. The KMP algorithm matches the characters txt[i] with pat[j] and increments i and j while pat[j] and txt[i] are matched. When a mismatch occurs, the followings hold:

- The characters pat[0..j-1] must match with txt[i-j…i-1]. (note that j starts with 0 and increments only when there is a match).
- lps[j-1] is a count of characters of pat[0…j-1].

From above two rules, we can conclude that one does not need to match these lps[j-1] characters with txt[i-j…i-1] because these characters must match with those in the pattern. The example of the KMP search algorithm is shown in Figure 2.



i=0, j=0
txt[] = "AAAAABAAABA"
pat[] = "AAAA"
lps[] = [0,1,2,3]
…
txt[i] and pat[] match, do i++ and j++.
i=4, j=4
Since j==M, **print pattern found** and **reset j**.(M represents the length of pat[])
j = lps[j-1] = lps[3] = 3
i=4, j=3
txt[] = "AAAAABAAABA"
pat[] = "AAAA"
lps[] = [0,1,2,3]
txt[i] and pat[] match, do i++ and j++.
i=5, j=4
Since j==M, **print pattern found** and **reset j**.(M represents the length of pat[])
j = lps[j-1] = lps[3] = 3
i=5, j=3
txt[] = "AAAAABAAABA"
pat[] = "AAAA"
lps[] = [0,1,2,3]
txt[i] and pat[j] do NOT match and j > 0, change only j.
j = lps[j-1] = lps[2] = 2
i=5, j=2
txt[] = "AAAAABAAABA"
pat[] = "AAAA"
lps[] = [0,1,2,3]
txt[i] and pat[j] do NOT match and j > 0, change only j.
j = lps[j-1] = lps[2] = 1
…
i=5, j=0
txt[] = "AAAAABAAABA"
pat[] = "AAAA"
lps[] = [0,1,2,3]
txt[i] and pat[j] do NOT match and j=0, do i++
i=6, j=0
txt[] = "AAAAABAAABA"
pat[] = "AAAA"
lps[] = [0,1,2,3]
txt[i] and pat[] match, do i++ and j++.
……

Figure 2: The searching process of the KMP algorithm.

- ROT13

ROT13 ("rotate by 13 places", also denoted by ROT-13) is a simple letter substitution cipher that replaces a letter with the 13th letter after it in the alphabet. ROT13 is a special case of **Caesar cipher** which was developed in ancient Rome. The ROT13 algorithm works by replacing the current English letters with its corresponding letter after 13 places. For example, in Figure 3, the letter A is replaced by N, B by O, C by P, etc. As a result, the word "HELLO" is replaced by "URYYB".

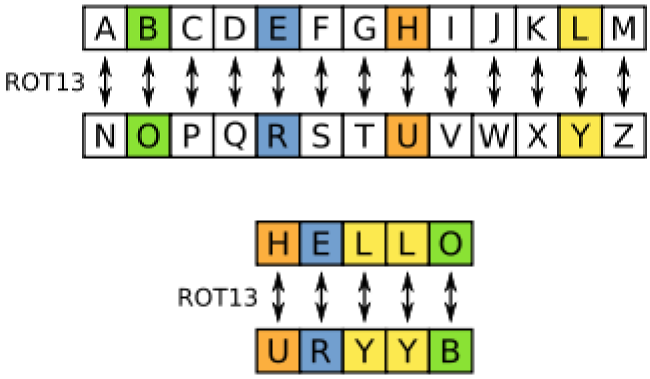


Figure 3: The ROT13 algorithm.

The formula to encode and decode ROT13 are shown as follows:

Encryption of a letter x by a shift n can be formulated in the following equation.

$$E_n(x) = (x + n) \mod 26$$

Similarly, decryption is formulated in the following equation.

$$D_n(x) = (x - n) \mod 26$$

- Implementation Steps

1. Read input data

Please write a program that can read data from standard input (stdin). Your program should be able to read multiple pairs of key-value data into an array. In this program, you would be asked to read three input data from a given file. The first two inputs are ***ciphertext*** and ***article***, respectively, and please store them as two separate char arrays. The third one is ***answer_dic***, which contains a set of key-value pairs with their corresponding secret phrases, and please store them as defined structures. To handle these input data, you should utilize **fgets** or **scanf** functions to read input data from the given file and store them as char arrays and defined structures. Both ***ciphertext*** and ***article*** can be processed respectively using a single **fgets.** However, ***answer_dic*** contains key-value pairs, and there are a total of 52 lines with the keys from uppercase A to lowercase z. To deal with ***answer_dic***, you will need to read data line by line. You should be careful about reading blank characters while reading standard input from the given file.

```
rnernvfgurguveqcynargsebzgurFhanaqgurbaylnfgebabzvpnybowrpgxa
bjagbuneobeyvsr
```

ciphertext

```
At the first of the beginning someone created the heaven and
the earea. The earea was without form, the earea is void; and
darkness was upon the face of the deep. And the Spirit of God
moved upon the face of the waters. And God said, Let there be
light: and there was light. And God saw the light, that it
was good: and God divided the light from the darkness.
```

article.txt

```
A,every plant
B,given you
C,every herb
D,oh god
E,which has
F,fruit producing
G,they will
.
.
```

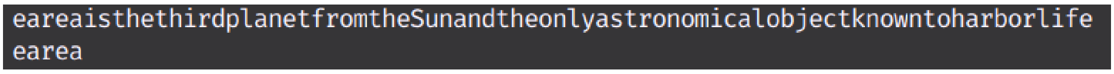
answer_dic
(partial content)

Figure 4: The required input data.

.

2. Using ROT13 to decrypt the secret string for finding the key

Please implement the ROT13 algorithm to decrypt ***ciphertext***. After completing the decryption, the first five characters are adopted as the key for finding the secret. Figure 5 shows the example of the above description. After decrypting the cipher text, the first five characters (**earea**) composes the key which can be used to compute the final answer later. The execution result is shown in Figure 5. Please output the cipher text and the key as shown in Figure 5.

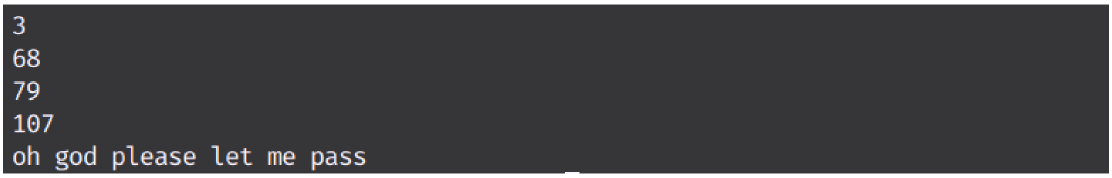


```
eareaisthethirdplanetfromtheSunandtheonlyastronomicalobjectknowntoharborlife
earea
```

Figure 5: Decrypt ciphertext and print out the required information.

3. Using KMP to search the pattern and find the secret message

Please implement the KMP algorithm to search the matched words with the key and find all the positions of the matched words in ***article***. Next, calculate the sum in the lps array (e.g., the sum of lps = {0,0,0,1,2} for key "**earea**" is **3**) and add the sum to the position of each matched word to form a **key index**. Finally, the last step is to map each key index with a corresponding ASCII character and the phrase in ***answer_dic*** to compose the secret message. Please output the results as shown in Figure 6. First, please print out the sum (**3**) of the lps array and the key indexes (**68**, **79**, **107**). After finding out all the phrases from ***answer_dic***, compose the phrases and print out in the same line (**oh god please let me pass**) without any punctuation.



```
3
68
79
107
oh god please let me pass
```

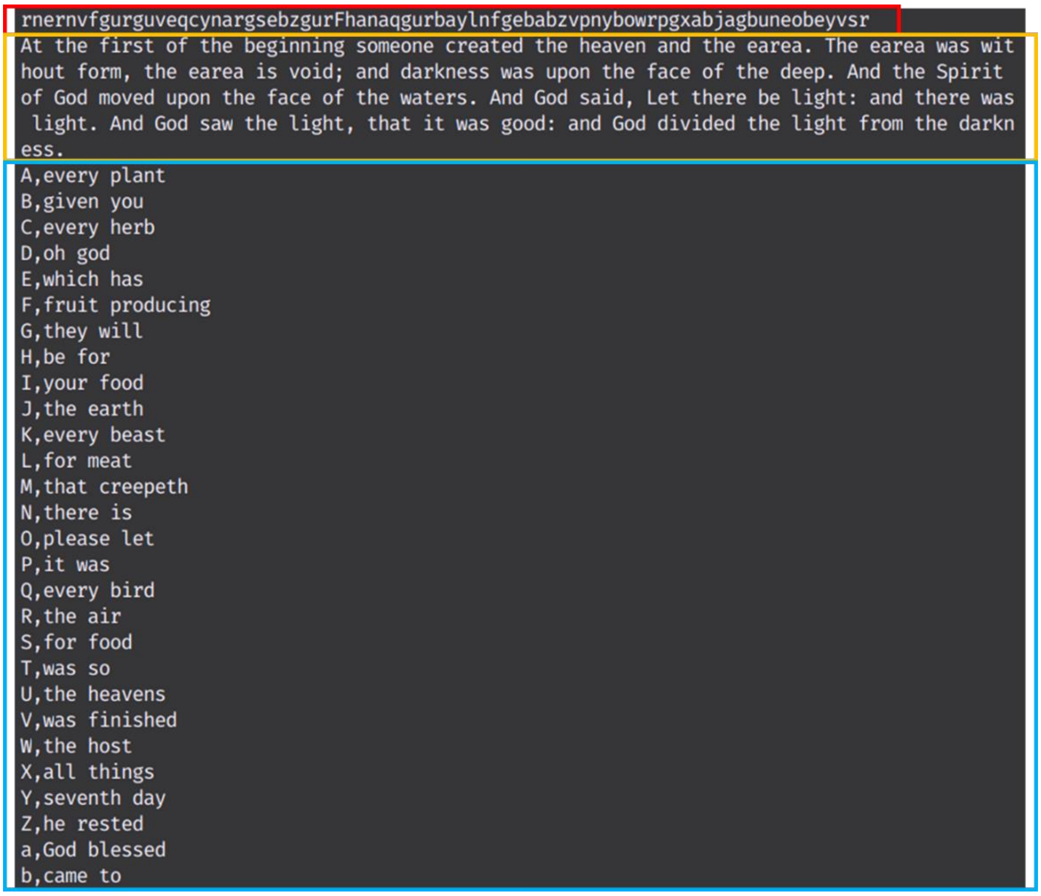
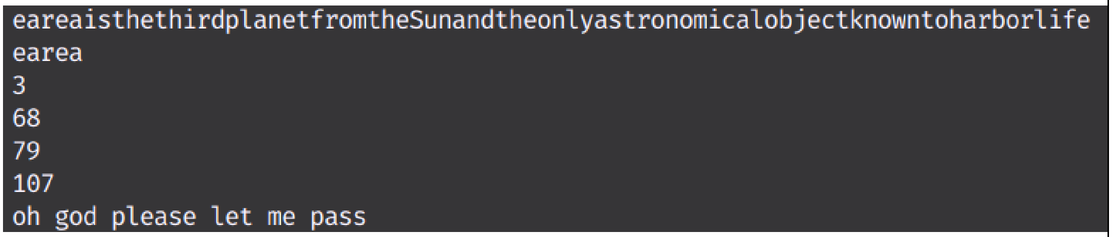
Figure 6. The process of the KMP searching.

Readme, comments and style (5%)

An indicator for good source code is readability. To keep source code maintainable and readable, you should add comments to your source code where reasonable. A consistent coding style also helps a lot when tracing the source code. For this assignment, please also compose a readme file in *.txt format and name it as "README.txt". This file should contain a brief explanation of how to use your program. Please remember to have your source code comments and readme file in English.

Test case examples:

The following input and output are the examples for testing your program. The format of your input and output should be exactly the same as shown in the following table.

Sample Input:	**Sample Output:**
Read input data. The input format is shown as follow: <ciphertext> <article> <answer_dic>	 rnernvfgurguveqcynargsebzgurFhanaqgurbaylnfgebabzvpnybowrpgxabjagbuneobeyvsr At the first of the beginning someone created the heaven and the earea. The earea was wit hout form, the earea is void; and darkness was upon the face of the deep. And the Spirit of God moved upon the face of the waters. And God said, Let there be light: and there was light. And God saw the light, that it was good: and God divided the light from the darkn ess. A,every plant B,given you C,every herb D,oh god E,which has F,fruit producing G,they will H,be for I,your food J,the earth K,every beast L,for meat M,that creepeth N,there is O,please let P,it was Q,every bird R,the air S,for food T,was so U,the heavens V,was finished W,the host X,all things Y,seventh day Z,he rested a,God blessed b,came to
Execution results. The output format is shown as follow: <Ciphertext> <Key> <sum of lps> <indexes> <secret message>	 eareaisthethirdplanetfromtheSunandtheonlyastronomicalobjectknowntoharborlife earea 3 68 79 107 oh god please let me pass

Submit

To submit your files electronically, login DomJudge website through the following url :

https://108-1-ds-judge.ate.cs.ccu.edu.tw/

Press the submit button and choose the homework questions you want to submit. After submitting your code, DomJudge will give you a result to tell you whether your code is correct or not. However, during the demo time, your code will be evaluated by different sets of test cases. Please make sure your code can work correctly based on the description above. Additionally, you must compress your code and the README file into a **zip** file and upload it to Ecourse2.

Grading policies

The TA(s) will mark and give points according to the following rules:

45% - Using the ROT13 algorithm to decrypt the ciphertext and find the key.
50% - Using KMP to search, find out the indexes, and the secret message.
5% - Readme, comments and coding style.