Assignment I — Structs and Arrays

TA: Tom (ga731852@gmail.com)

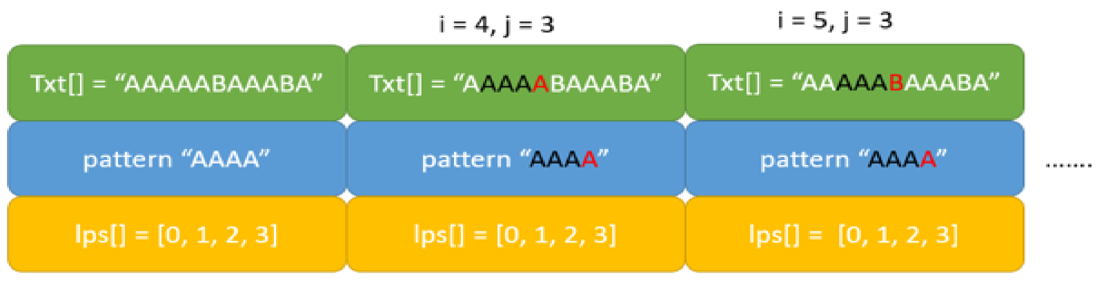
Deadline: 2018 Oct. 9, 11:59pm

Structures and Arrays

In this homework, your will be asking to implement the KMP algorithm. Pattern searching is an important problem in computer science. When we do search for a string in a word file, a website, or a database, a pattern searching algorithm is used to show the search results. 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[] of size m (equal to the size of the pattern) which is used to skip the number of characters while matching. The details about lps[] can be referred to the KMP algorithm in the thextbook.
Following are examples of lps[].

pattern "AAAA"	pattern "AAABAAA"	pattern "AABAACAABAA"
lps[] = [0, 1, 2, 3]	lps[] = [0, 1, 2, 0, 1, 2, 3]	lps[] = [0, 1, 0, 1, 2, 0, 1, 2, 3, 4, 5]

After constructing lps[], we will use the value from lps[] to determine the next characters to be matched. When we find a mismatch. We know that characters pat[0..j-1] match with txt[i-j+1…i-1] and lps[j-1] represents the count of characters of pat[0…j-1]. Therefore, we can use lps to determine the next match position. Following are KMP algorithm examples.



txt[i] and pat[j] match , do i++, j++

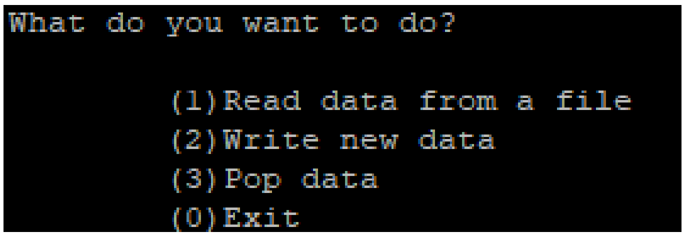
Since j == M, print pattern found and reset j, j = lps[j-1] = lps[3] = 3

txt[i] and pat[j] do NOT match and j > 0, change only j j = lps[j-1] = lps[2] = 2

M denotes the value of lps[x]

Action menu

Because the program is meant to be used by a user, who does not want to modify source code, please make your program user-friendly. You must add an action menu. When the program is being executed, a question with multiple options will be shown. The Figure1 is an example for the action menu:



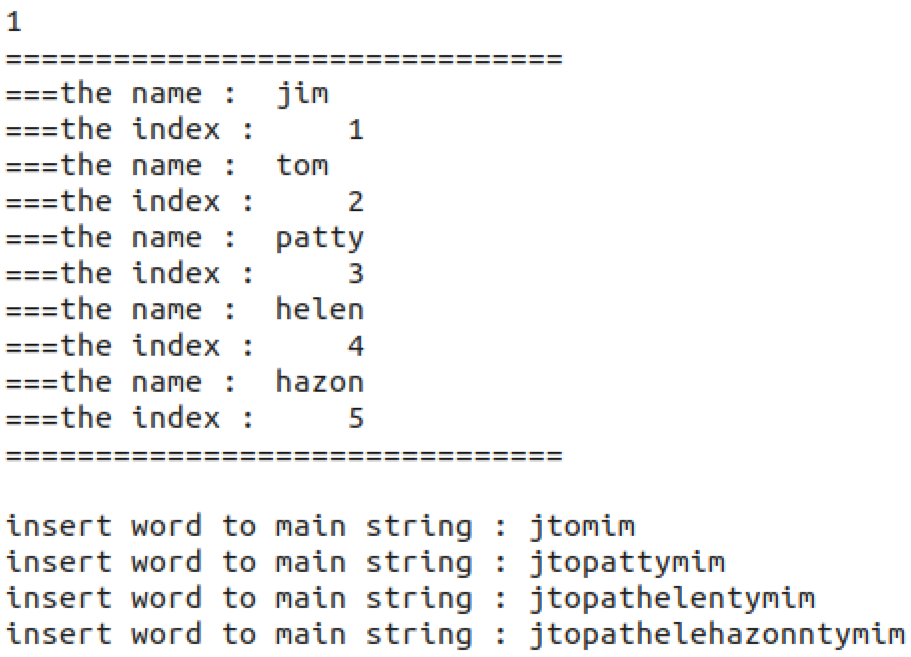
```
What do you want to do?

        (1)Read data from a file
        (2)Write new data
        (3)Pop data
        (0)Exit
```

Figure 1. An illustration of an action menu.

The user enters a desired option (i.e., 0, 1, 2 or 3) and the program executes the function. For an action menu, you can use this piece of code as a template. The following sections explain the detail for each function.

1. Read data from a file

Please write a program that can read data from a file and declare a char array to store the data into a defined structure. The foregoing process is shown in Figure 2.



```
1
===============================
===the name :  jim
===the index :     1
===the name :  tom
===the index :     2
===the name :  patty
===the index :     3
===the name :  helen
===the index :     4
===the name :  hazon
===the index :     5
===============================

insert word to main string : jtomim
insert word to main string : jtopattymim
insert word to main string : jtopathelentymim
insert word to main string : jtopathelehazonntymim
```

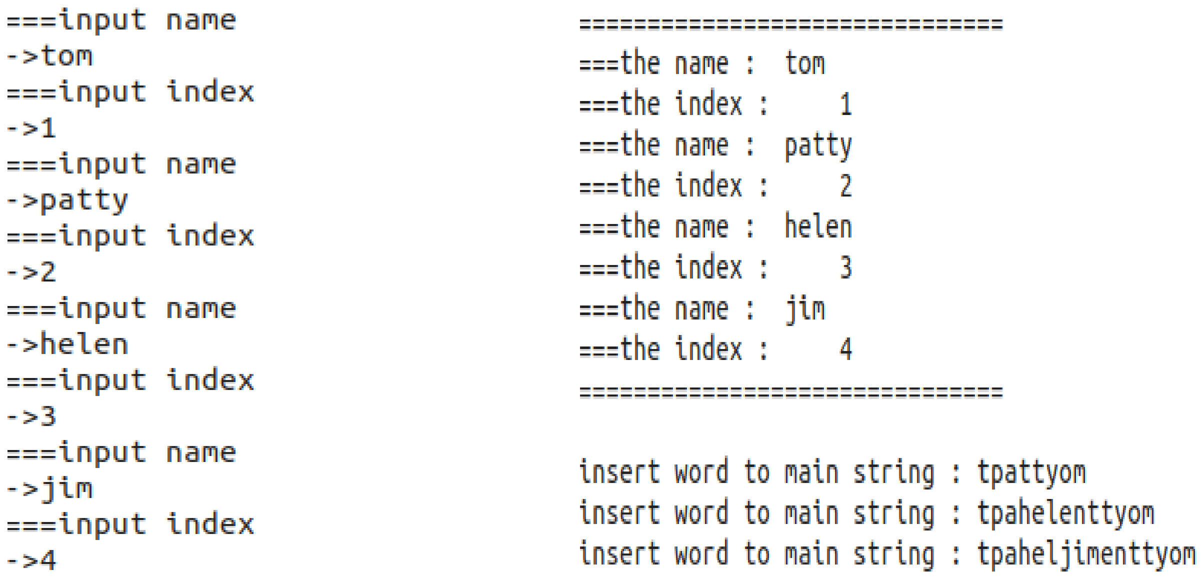
Figure 2. An illustration of reading a file.

Your program should be able to read multiple pairs of large numbers into an array. To make the records better usable for the researchers, we want to allow using third party products, such as Microsoft Excel or LibreOffice Calc to work with our program. Luckily, these programs can read file in CSV format. Please let your program operate with CSV files. In our tests, you can expect that there will be exactly 5 pairs of data. The records will be given in the csv-format as below.

```
name1, index1
name2, index2
name3, index3
name4, index4
name5, index5
```

2. Write data to a structure

In this section, please program the function which can write data to a structure. please input the name and index from the command line. The value of index represents the index where the next word to be inserted. Therefore, you will append the structure data to the main string and print it in the commend line. Figure3 illustrates the procedure of adding data to the main string.



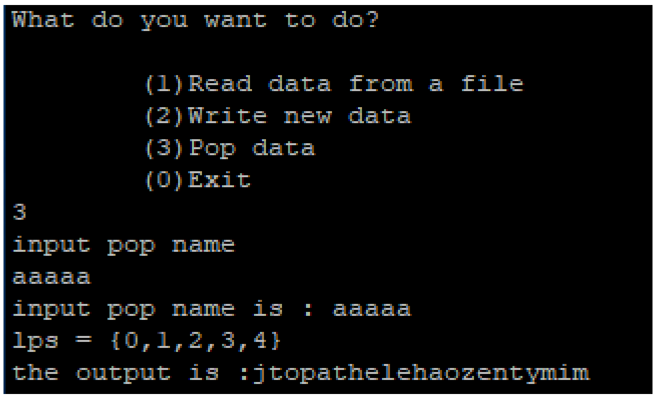
```
===input name
->tom
===input index
->1
===input name
->patty
===input index
->2
===input name
->helen
===input index
->3
===input name
->jim
===input index
->4
```

```
===============================
===the name :  tom
===the index :     1
===the name :  patty
===the index :     2
===the name :  helen
===the index :     3
===the name :  jim
===the index :     4
===============================

insert word to main string : tpattyom
insert word to main string : tpahelenttyom
insert word to main string : tpaheljimenttyom
```

Figure 3. An illustration of writing data to a file.

3. Using KMP to search data form the main string and pop out the data

After reading or inputting data to the main "text string" Txt. Please program the KMP algorithm so that a user can execute program to search match data and pop it. In this part, you should let the user input a string as a match pattern. Print out the remaining strings after each pop and the value of lps[]. The above description is shown in Figure 4.



```
What do you want to do?

        (1)Read data from a file
        (2)Write new data
        (3)Pop data
        (0)Exit
3
input pop name
aaaaa
input pop name is : aaaaa
lps = {0,1,2,3,4}
the output is :jtopathelehaozentymim
```

Figure 4. An illustration of popping out data from the main string.

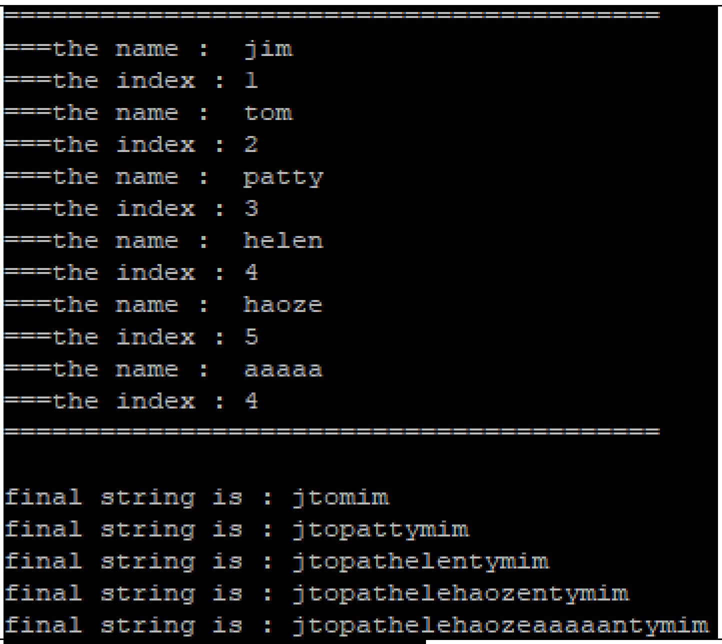
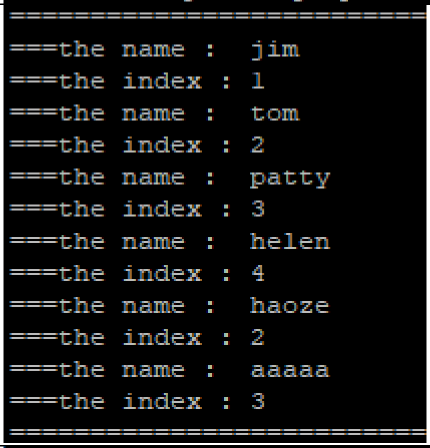
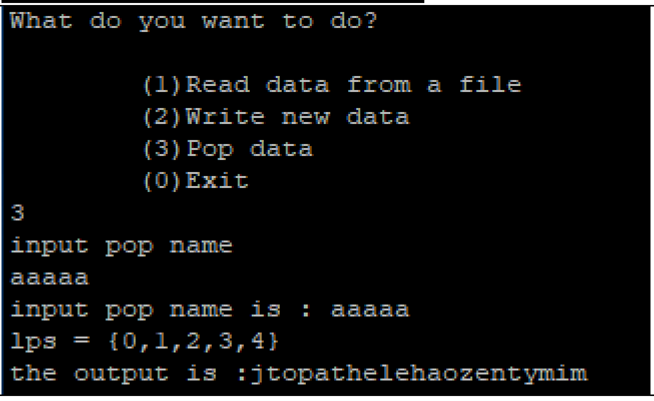
4. Readme, comments and style (5%)

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.

5. Test case examples:

The following inputs 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:**

From the menu, choose: 1 (Read data from a file)	 == ===the name : jim ===the index : 1 ===the name : tom ===the index : 2 ===the name : patty ===the index : 3 ===the name : helen ===the index : 4 ===the name : haoze ===the index : 5 ===the name : aaaaa ===the index : 4 == final string is : jtomim final string is : jtopattymim final string is : jtopathelentymim final string is : jtopathelehaozentymim final string is : jtopathelehaozeaaaaantymim
From the menu, choose: 2 (Write data to a structure)	 ========================== ===the name : jim ===the index : 1 ===the name : tom ===the index : 2 ===the name : patty ===the index : 3 ===the name : helen ===the index : 4 ===the name : haoze ===the index : 2 ===the name : aaaaa ===the index : 3 ==========================
From the menu, choose: 3 (Search data form main string and pop data)	 What do you want to do? (1)Read data from a file (2)Write new data (3)Pop data (0)Exit 3 input pop name aaaaa input pop name is : aaaaa lps = {0,1,2,3,4} the output is :jtopathelehaozentymim

6. Submit

To submit your files electronically, enter the following command from the csie workstation:

```
turnin DS_I_2018.hw1 [your files…]
```

To check the files you turnin, enter the following command from the csie workstation:

```
turnin –ls DS_I_2018
```

You can see other description about turnin from following link:
https://www.cs.ccu.edu.tw/lab401/doku.php?id=turninhowto

7. Grade policies

The TA(s) will mark and give points according to the following rules:
10% - Read data from a file e and store them into a structure.
15% - Show the read data.
 5% - Action menu.
50% - Using KMP to search and pop a match word.
15% - Write data to a structure and print it out.
 5% - Readme, comments and coding style.