ENGR 14200
C 5 PA: Pointers & Arrays in C

Individual Assignment: See the course syllabus for a definition of what constitutes an individual assignment.

Task 1 (of 2)

Learning Objective: Practice using pointer notation to access arrays.

Background: As part of your research internship, you have been tasked with analyzing the results of 500 different Infrared Spectroscopy tests on the same compound. Infrared spectroscopy uses the functional groups present in the sample to identify the compound. The professor in charge of the lab has been working to develop a machine that will accurately identify the identity of the substance from the peaks found. He wants to make sure that the instances in which the system correctly identifies the substances are readily recognized. He wants to ensure that you have sufficient programming skills to perform the intended task. He has suggested that the system alter the incorrect identifications by reversing the order of the characters and that the system print "That's right!" for all appropriate identifications. He would like a prototype of this program.

Develop an algorithm using a high and low level flow chart and then write a program that reads a string of up to 15 characters with no embedded blanks from a user input, uses a function to reverse the string, and prints the reversed string. However, if the string entered is "ENGR-awesome," the program should instead print, "That's right!" Save your flow chart as `C5_PA_login.pdf`

Save your program as: `C5_PA_Task1_login.c`

Sample Input/Output Format:

```
Enter a string: STAR
Output string: RATS

Enter a string: ENGR-awesome
Output string: 'That's right!'
```

Requirements and Hints:
In the function **main**, the user is prompted for the character string. Your program should read the string, then invoke a function named `revcheck()` that will reverse and check the string. To check the input string, you may individually compare each character or use `strcmp`. The desired string is then displayed back in **main**.

To reverse the input string, the function `revcheck()` reverses the string in place and uses pointer syntax. It accepts a single argument: the address of the first cell in the array of characters. Inside the function `revcheck()`, two pointers are used: `s`, which initially points to the first character, and an auxiliary pointer `end`, which initially points to the last character (the character just before the null terminator, which is left in place). As long as `s` is `less` than `end`, swap the characters to which `s` and `end` point, increment `s`, and decrement `end`. You may **not** use `strlen()` in this task.

Task 1 Files:

1) `C5_PA_login.pdf`
2) `C5_PA_Task1_login.c`

Task 2 (of 2)

Learning Objective: Practice using pointer notation to access arrays.

Background: During times of war, it is very important that intelligence not be leaked to the opposition. To combat this leak, many nations developed encryption techniques to secure any messages they wanted to send over long distances. One example of a technique is using transposition ciphers. Transposition ciphers take a scrambled message and reorganize it to form the message using matrices of sizes indicated by a keyword.

Instructions:
For this task, you will create a program that takes a keyword as input and uses that keyword to unscramble a string of randomized characters using pointers.

You will first prompt the user to input a keyword of 20 or fewer characters. The user must first determine the number of non-repeating characters in the keyword. For example, the word `Franklin` has 7 non-repeating characters. Perform this task in a function called `repeating()`, which should take a pointer as input.

Next, using a function called `groupSize()`, which takes a pointer and the number of non-repeating characters as inputs, assign numerical values to each of the letters based on their position in the alphabet (A=1, B=2, …). In the same function, determine how many letters are in each group of characters, using the formula:

$$groupSize = \left(\sqrt{\frac{Sum\ \ of\ \ characters}{Number\ \ of\ \ nonrepeating\ \ characters}} \right)$$

The remainder that results from this function should be truncated so that the group size is a positive, integer value. The calculated value indicates the number of letters that are in each group. For example, if the keyword is `Franklin`, the sum of all of the characters is 85 and the number of non-repeating characters is 7. Thus, there are 3 characters per group.

Next, the user will input a long string of scrambled characters of no more than 1000 characters. This is the message to be deciphered. The total number of characters in the string should be equal to the number of groups multiplied by the number of characters in each group. If the number of characters does not match this number, the excess characters can be ignored, and the program should continue to decipher the message. If there are excess characters, when outputting the final deciphered message, the program should also state that the code may have been altered by enemy spies. You may not use `strlen()` in this task

The string to be deciphered should be split into the proper number of groups with the proper number of characters in each group. The number of groups is equal to the number of non-repeated characters in the keyword. Each group of characters should be a single array of characters within a larger array. The order of the smaller arrays relates to the order of the letters in the keyword, keeping only the first instance of any repeating letters. In order to decipher the message, the smaller arrays should be swapped in place using pointers, such that the arrays should be put in alphabetical order according to the corresponding character from the keyword.

Hint: Don't forget function prototypes. Develop an algorithm for the program in the form of a high and a low level flow diagram before you beginning programming. Save your flow diagram in your previously created PDF. Translate your flow diagram to a code in C. Save the program as: `C5_PA_Task2_`*`login`*`.c`

The overall program inputs and outputs should look like (inputs in bold):

```
Input a keyword: FRANKLIN
Input the message to be decoded: VESWARIHAATEWAR.IHEEN

Decoded message: IHA VES EEN WAR .IH ATE WAR
```

The output was determined as follows:

1. The function `repeating()` determines that there is 1 repeating letter in the keyword, such that there are 7 non-repeating keywords. Thus, there should be 7 groups of letters
2. The function `groupSize()` determines that there are to be 3 letters in each group.
3. There should be 21 characters in the message string. This is true. No error message needs to be included.
4. The array of arrays should be the following: `VES WAR IHA ATE WAR .IH EEN`
   This array was determined as follows:

F	R	A	N	K	L	I	N
VES	WAR	IHA	ATE	WAR	.IH	EEN	(repeated character)

5. The order of the arrays in the array of arrays should be swapped such that the corresponding characters in the keyword are in alphabetical order. Thus the array has the following entries:
   `IHA VES EEN WAR .IH ATE WAR`
   This array was determined as follows:

A	F	I	K	L	N	R
IHA	VES	EEN	WAR	.IH	ATE	WAR

Thus, the decoded message is “I have seen war. I hate war,” which is a quote from President Franklin Delano Roosevelt’s address at Chautauqua, N.Y. in 1936.

Task 2 Files:

1) C5_PA_*login*.pdf
2) C5_PA_Task2_*login*.c

Make sure you include your INDIVIDUAL HEADER in your file(s) and submit ALL files to the appropriate turn-in box on Blackboard.