CSC 362 Programming Assignment #3
Due Date: Monday, March 13

In this assignment, you are to create a program that will encrypt and decrypt messages. You will input the initial message from a text file and output an encrypted version of the message to the console window and then a decrypted version of the message (which should be the same message as the original input message) to the console window.

The encryption code is a simple rotation algorithm, adding or subtracting an int value to each *letter* in the message (non-letters remain unchanged). The code for all letters is given by the values in the int array {5, 2, -1, 2, -3, 2, -4, -2, 6, 4}. Each number indicates how much to add or subtract from the letter to encode or decode it. For instance, the first letter will have 5 added to it (an 'A' becomes an 'F', a 'p' becomes a 'u'), the second letter will have 2 added to it and the third letter will have -1 added to it. As adding or subtracting to a letter may change it from a letter to non-letter, wrap the values around if you go beyond the end of the alphabet (for instance, if adding 5 to 'Z', rather than it becoming '_' it should be wrapped around to be an 'E'). If the message starts with "The", this becomes "Yjd" ('T' + 5 = 'Y', 'h' + 2 = 'j' and 'e' – 1 = 'd').

To rotate a letter, add the rotation value (as found in the array above) to the current character in the message and cast the result back to a char (adding changes the chars to ints). Assume your message is stored in `char str[5000]` and the code is stored in `int x[10]`. If you are working on the ith character of str and you have already processed j-1 previous letters, then the instruction would be `str[i] = (char) (str[i] + x[j]);` To perform the "wraparound", if the character was an upper case letter and is now > 'Z' then subtract 26 and if the character was a lower case letter and is now < 'a', add 26 to it, etc. Make sure you retain the proper case (upper case letter becomes an upper case letter, lower case letter becomes a lower case letter). NOTE: you might want to use the ctype library functions of isalpha, isupper and islower.

This program sounds pretty easy right? Here's the catch: you can *only* access the two arrays (the string and the int array) through pointers and you must move through the arrays using pointer arithmetic. You will need both a `char *` to point into the message and an `int *` to point into the encryption array. To move through the character array, use a loop that ends once the current character is '\0'. Only convert letters, non-letters remain unchanged. The int array is 10 numbers long and you only use it for converting letters. After converting a letter, move to the next position in the int array but do not increment the int array pointer on a non-letter. As an example, if your message is "Hi! W8t 1." and you are working on the 't' character (at location 6 of the str array), you would only be at x[3] of the int array because you would not have used x for the '!', the space or the '8'. Once your int array pointer has reached the end of the int array, it must be reset to point at the beginning of the array. You can do this by resetting the pointer back to x or &x[0]; NOTE: do not hardcode the number 10 in the program as the size of the int array. If you change that int array's size, you would have to change all of the hardcoded 10s. Instead, when you pass the int array to a function, also pass its size. You can initialize a size variable in main to 10 or a constant defined through a #define.

The program must be broken into a main function and the following auxiliary functions:

- Input the file character by character into a char array, adding \0 at the end of the array (\0 will not be at the end of the input file). The message will be no longer than 4999 characters (the array will need 5000 characters so that you can include the \0). Pass the char array to this

function. Since the char array is an address, there is nothing for you to return from this function.

- Encrypt the message by passing this function a pointer to the beginning of the message (string), a pointer to the beginning of the encryption array (the int array), and the size of the int array. The encrypt function will encrypt the original char array, not a copy. The function will iterate through the array until it reaches the \0 character. To reset your int array pointer back to the beginning of the int array, you should make a copy of this pointer. For instance, if this array is passed in as x and your local parameter is ip, you can reset it to the beginning by doing `ip = x;` or `ip = &x[0];` or even `ip = ip - n;` where n is the size of the array.
- Decrypt the message passing this function a pointer to the beginning of the message, a pointer to the beginning of the encryption array, and the size of the encryption array. This function is identical to encrypt except that you will change + to – and – to +. You may use a single function to both encrypt and decrypt the message if desired, in which case you would pass a fourth parameter specifying whether you are encrypting or decrypting.
- An output function to receive a pointer to the beginning of the message (as a char *) and output the message character-by-character to the monitor, using putchar, until you reach \0. I am disallowing the use of printf for this assignment.

Your main function must call each of the above functions in turn. You may write all of your functions in a single file. You may use a header file but you do not have to. All output goes to the console window.

Run your program on the first test input file on my website (message1.txt) and compare your results to the output below. Once successfully running, run your program on the second input file on my website (message2.txt) and collect a screen capture of the output. Hand in a hardcopy of your commented program and the output (both the encrypted message and the decrypted message) from running your program on the second file (message2.txt). REMEMBER: only access arrays using pointers, and adjust your pointers using pointer arithmetic.

```
Encrypted message:
Npeqoowross kr plv gluaqgcib,
mjmcpjffg fu jmz anucqj,
yeqjsr kr plv ppaxm,
vqwqj eq tsy ddcrvu,
zkezvx kp pkr rsag,
kqsg eq tsy otufe,
ISYMH kr veg XCYX!

Decrypted message:
Information is not knowledge,
knowledge is not wisdom,
wisdom is not truth,
truth is not beauty,
beauty is not love,
love is not music,
MUSIC is the BEST!
```