Task:

Write a C program that conforms to the requirements listed below.

Requirements:

* You are building a magic decoder ring; a program capable of reading lines of text from a file and encrypting it; or reverse the process by reading an encrypted file and decrypting it
* Encryption and decryption require you to follow a specific set of steps to encrypt, or a reverse set of steps to decrypt
* For this program we will combine the use of bit masks and bitwise Exclusive Or operations to produce the results.
* User Interface: Prompt the user prompt for an operation – either Encrypt or Decrypt, and then for a text string, capture the input, and then do the requested action. See encryption, below.
* Set up an array of printable characters to use in a substitution cipher as one step of the encryption:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Start** | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| **Sub** | : | ; | ? | . | , | / | 0 | 1 | 2 | 3 |
| **Start** | A | B | C | D | E | F | G | H | I | J |
| **Sub** | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D |
| **Start** | K | L | M | N | O | P | Q | R | S | T |
| **Sub** | E | F | G | H | I | J | K | L | M | N |
| **Start** | U | V | W | X | Y | Z | ! | @ | # | $ |
| **Sub** | O | P | Q | R | S | T | U | V | W | X |
| **Start** | % | & | ( | ) | : | ; | ? | . | , | / |
| **Sub** | Y | Z | ! | @ | # | $ | % | & | ( | ) |

* + Note the table uses capitals. Either force entry of the characters in capitals or perform a toupper function for each element of the character array.
* Set up an enumerated type with values of ENCRYPTING and DECRYPTING to control the direction of the process based on the user’s choice.
* Set up an encryption bitmask, called mask with this value: unsigned int mask = 0xa5;
* Set up two unsigned int arrays and a character array, each of size 80. These will hold the character or numeric sequences
* Set up two pre-processor macros, ENCRYPT and DECRYPT to perform the bitwise Exclusive Or. Each should take one parameter, the input character sequence of each line converted to an integer. You would then use the macro with something similar to:

Where x[] is an array holding the result:

* + - x[idx] = ENCRYPT(numeric\_text[idx]); for each character in the encrypted text.
* Encryption:
  + Perform the letter offset substitution cipher from the table on all characters (non-blank entries) in the input text
  + Convert each character in the substituted text to type integer. The easiest way to do this is to create a corresponding integer array and use a cast to perform the conversion of type and copy and convert each character to an entry in this integer array, including spaces
  + Encrypt the integer representation of the characters in the integer array using the ENCRYPT macro. This includes all of the space characters as well
  + Print out the encrypted result using decimal format so you can see the numeric value of each encoded character. Cypher groups will look something like:
    - 3345 3123 5453 345 323 147 312 and so on
    - Each space delimits a new character (because space characters are now encrypted as their own number sequence)
* Decryption
  + Reverse the encryption sequence on the encrypted text to decrypt it. Remember to capture the decrypted entries in decimal (using the space delimiters for each “character”) and store them in an integer array to convert back to text
  + Print out the decrypted text to a file. Confirm it matches the original text from the first (plaintext) input file.
* Don’t forget to reset the character array between lines to keep things synchronized.
* Finally, create a file containing these phrases:
  + THE QUICK BROWN FOX JUMPED OVER THE LAZY DOG
  + WATSON; COME HERE AT ONCE!
  + I’LL MEET YOU AT 3:00 O’CLOCK. #SEEYOULATER
* Run the file through the encryption and create an encrypted file.
* Run the encrypted file back through decryption to ensure it works.
* Try your own message file, and trade message files with your friends as a further test.
* Maintain a standard layout/format for the code. Be consistent with spacing or tabbing, use the layout o make nested operation visually clear.

Evaluation:

This assignment is worth 22 marks. Please see the marking rubric below.

Assignment Notes:

The assignment must be demonstrated to the instructor on or before the due date during class.

If your assignment is late please send an e-mail to the instructor, hal.o’connell@nscc.ca, to confirm submission. This e-mail will constitute the timestamp for evaluating any late penalty the assignment may incur.

See the **Marking Rubric** below.

<https://www.rosipov.com/blog/c-strtok-usage-example/>

| Criteria | Below Standard | Developing | Acceptable | | Professional | Marks |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | | 3 |
| Macros | * Macro(s) not used | * Macro(s) used but contains more than 2 errors | * Macro(s) used and almost matches specification but has an error | | * Macros used correctly with no bugs | \_\_\_\_\_\_ |
| Bit Operations | * XOR not used | * XOR used but more than 2 errors | * XOR work mostly as expected but still one error | | * XOR implemented completely to specification | \_\_\_\_\_\_ |
| Enumerated Types | * Not attempted | * used but more than 2 errors | * work mostly as expected but still one error | | * implemented completely to specification |  |
| Input File Operations | * Files not used | * Files used but more than 2 errors | * Files work mostly as expected but still one error | | * File support implemented completely to specification |  |
| Output File Operations | * Files not used | * Files used but more than 2 errors | * Files work mostly as expected but still one error | | * File support implemented completely to specification |  |
| Encryption | * not attempted | * Substitution cypher and XOR attempted but several errors | * Substitution cypher and XOR attempted but minor error(s) | | * implemented completely to specification | \_\_\_\_\_\_ |
| Decryption | * not attempted | * Substitution cypher and XOR attempted but several errors | * Substitution cypher and XOR attempted but minor error(s) | | * character table(s) implemented completely to specification | \_\_\_\_\_\_ |
| Aesthetics of Output | * incorrect or non existent use of whitespace in output * output is confusing and hard to follow | * fair use of   whitespace   * most output is clear, but poorly presented | * good use of whitespace * output is clear and well presented | | * excellent use of whitespace   output is clear and attractively presented | \_\_\_\_\_\_ |
| Readability | * source code does not match specification | * source code is exceptionally well organized and easy to follow |  | |  | \_\_\_\_\_\_ |
| Comments | * comments do not match specification | * not over/under commented * comments are meaningful and easily understood * files and functions have headers * Code is self-documenting |  | |  | \_\_\_\_\_\_ |
| **Naming** Convention | * no standard naming convention followed | * industry standard naming convention used throughout the program |  | |  | \_\_\_\_\_\_ |
| Consistency | * no consistency in formatting or layout of source code | * source code formatting never deviated from expected layout |  | |  | \_\_\_\_\_\_ |
| Total | | **29** |

0 - Assignment not submitted or work not original.