ECE 3331, Fall 2023, Program 05, Due Sunday 09/24 at 11:59 pm on Blackboard

You will write a program that can decode an encoded text message.
The encoded message will be located in an input text file called **input.txt** , located in the directory from which your program is run.
Your program will store the decoded message in an output text file **output.txt** , also located in the directory from which your program is run.

Your main program will read the input file in.txt from the same directory within which your executable is run, and generate the output file out.txt in that same directory.

The input file can contain characters spanning the full range of a 1-byte character (0-255), but only the characters with asci codes 33-71 (inclusive of 33 and 71) contain the original text message. Your program will read-in the characters from input.txt, decode the characters with asci codes 33-71, and **skip-over** any characters with asci codes outside the range 33-71.

How to decode the message
The decoded message will only consist of the lowercase alphabet characters, the digit characters 0-9 , the blank space character, the period, and the newline.
The ascii character codes for the allowable characters in the decoded message are
97-122 for the lowercase letters,
48-57 for the digit characters 0-9,
ascii code 32 for the blank space,
46 for the period, and 10 (plus the Carriage Return 13 that is discarded by the operating system when reading the \n) for the newline.
Therefore, there are a total of 26+10+3=39 allowable characters (a-z, 0-9, blank, period, LF) in the decoded text message.

In the encoded text message in input.txt, these 39 ascii character codes were encoded to the 39 characters with ascii codes 33-71 (inclusive of 33 and 71) as follows.
The lowercase alphabet a-z were encoded as ascii character codes 33-58.
The digit characters 0-9 (asci codes 48-57) were encoded to ascii character codes 59-68,
and the blank, period, and LF were encoded to ascii character codes 69, 70, and 71 respectively.
That is, in the encoded text message, character 'a' (asci code 97) was encoded as character '!' (asci code 33).
Character 'b' (asci code 98) was encoded as the double quotes character “ (asci code 34).
Etc…

Any other ascii characters in the input file input.txt are not considered as part of the message and will be skipped-over by your program.
Again, to decode a file, you will ignore any characters in the input.txt that are not asci codes 33-71. You will only decode the characters in input.txt if they are within the range 33-71, and you will store the decoded characters one-by-one in the output file output.txt.

Project requirements for your ANSI C code:

1. Your main( ) function will open the two files (input.txt and output.txt).
2. Your main( ) function will use fscanf( ) in a loop to read-in characters from file input.txt until fscanf( ) returns EOF. Note, fscanf( ) will return the integer EOF when it tries to read-in another character from input.txt, but there are no more characters in the file to be read.
3. After a character is read-in from input.txt, your main( ) function will call a subroutine to decode the character. The subroutine will take one parameter, the character to be decoded, and the subroutine will return an integer 0-255 that is the asci code for the character it decodes. If the character sent to the subroutine is outside of the range 33-71, the subroutine will return the integer -1. Use fprintf( ) to write the decoded characters into the file output.txt
4. Your subroutine must use the **switch( )** statement to determine the ascii character code to print to the decoded file output.txt. Important: don't include a separate case in the switch( ) function for each and every possible input/output char value. That would require too many lines of code. Be smart! Note that the lowercase characters with ascii codes 97-122

were encoded as ascii codes 33-58. Therefore, to decode an encoded character (ascii codes 33-58), your subroutine should simply add 64 to the encoded ascii code.
To decode input ascii codes 59-68 (encoded digit characters 0-9 with ascii codes 48-57), your subroutine should simply subtract 11 from the encoded ascii code.
The encoded blank space, period, and newline can be decoded in similar fashion.
Therefore, your switch( ) statement only needs 5 cases.
Declare an int flag in your subroutine.
If the asci code of the input character is 33-58, set the flag =1 (case 1: decodes a char a-z) .
If the asci code of the input character is 59-68, set the flag =2 (case 2: decodes a digit char 0-9) .
If the asci code of the input character is 69, set the flag =2 (case 3: decodes a blank space) .
If the asci code of the input character is 70, set the flag =2 (case 4: decodes a period) .
If the asci code of the input character is 71, set the flag =2 (case 5: decodes a newline) .
If the asci code of the input character is outside the range 33-71, you must ensure that your subroutine returns the int -1 to the main( ) function.

Below are two examples of encoded and decoded files.

Now, to decode the file
!"#G!"#G
the decoded file will be
abc
abc

To decode the file
!"#G!"#G
the decoded file will be
AaBbCc
aAbBcC

To test your program, there are two are the two input.txt file examples above, but they are named input1.txt and input2.txt.