Mystery Readme

The first thing I figured out about mystery was the input it takes. The format to run it without getting an invalid argument was to run this line in the terminal after compiling it:
./mystery Input:yourstringgoeshere

There are 3 different kinds of inputs this program takes.
If all of the letters are lowercase, the program produces an output string with the pattern:
lllUlUlUlUl..... (where l is lowercase letter and U is uppercase letter).
This pattern is the first 3 letters are lowercase, then it alternates between upper and lower case.

If all of the letters are Uppercase, it gives an output of:
UllUlUlUUUlUlUUUlUl.....
one thing to notice about this pattern that is unique is that it is lower casing all of the letters that are prime numbers in the string, except for the first one which stays upper case.

Now, the most compilcated part of this is figuring out what happens if there is a mixture of uppercase and lowercase letters in the string. What happens in the program is there a counter variable (for example purposes I will call it count) telling the program what number letter they are on in the input. There is another check to see if the last letter(s) behind it matched cases. For example if there is a string of 2 consecutive lowercase letters, the program then has the string follow the pattern of lllUlUlUlUl..... but as soon as it switches over to an uppercase letter, the output string then checks if count is prime, if it is the letter is lowercase, if not it stays uppercase, and this is put into the output string until the very end where it reaches the end of the string and gives you the final output.

The big O notation of this method is O(n) to go through the string and check for the patterns to output. The method to check for prime is O(n) so the total runtime of the program is O(n^2). The space used by this program is just simply the size of a char multiplied by the length of the string because after the input is used it can be freed and the output is the same length of the input, so really throughout the program the most space that is used at one time is 2*(sizeof(char) * lengthofstring).