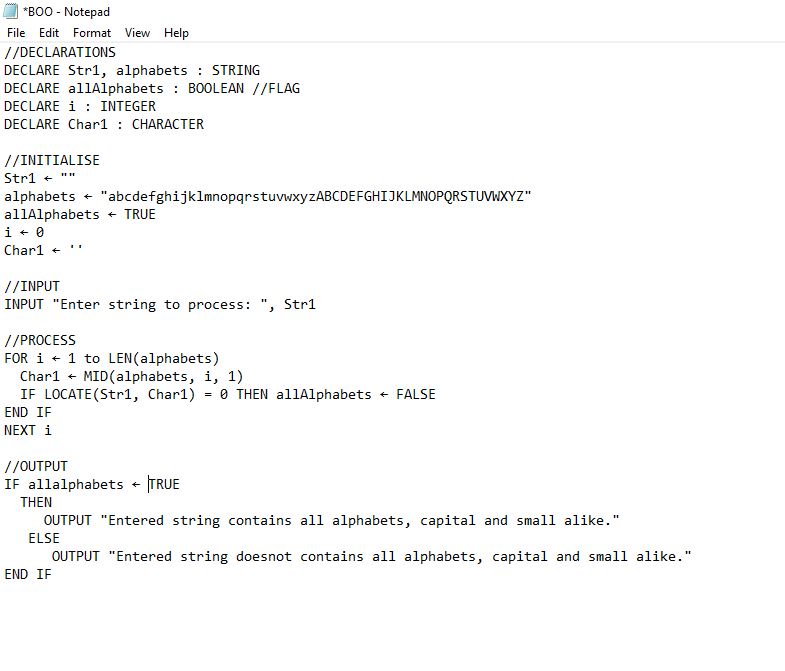
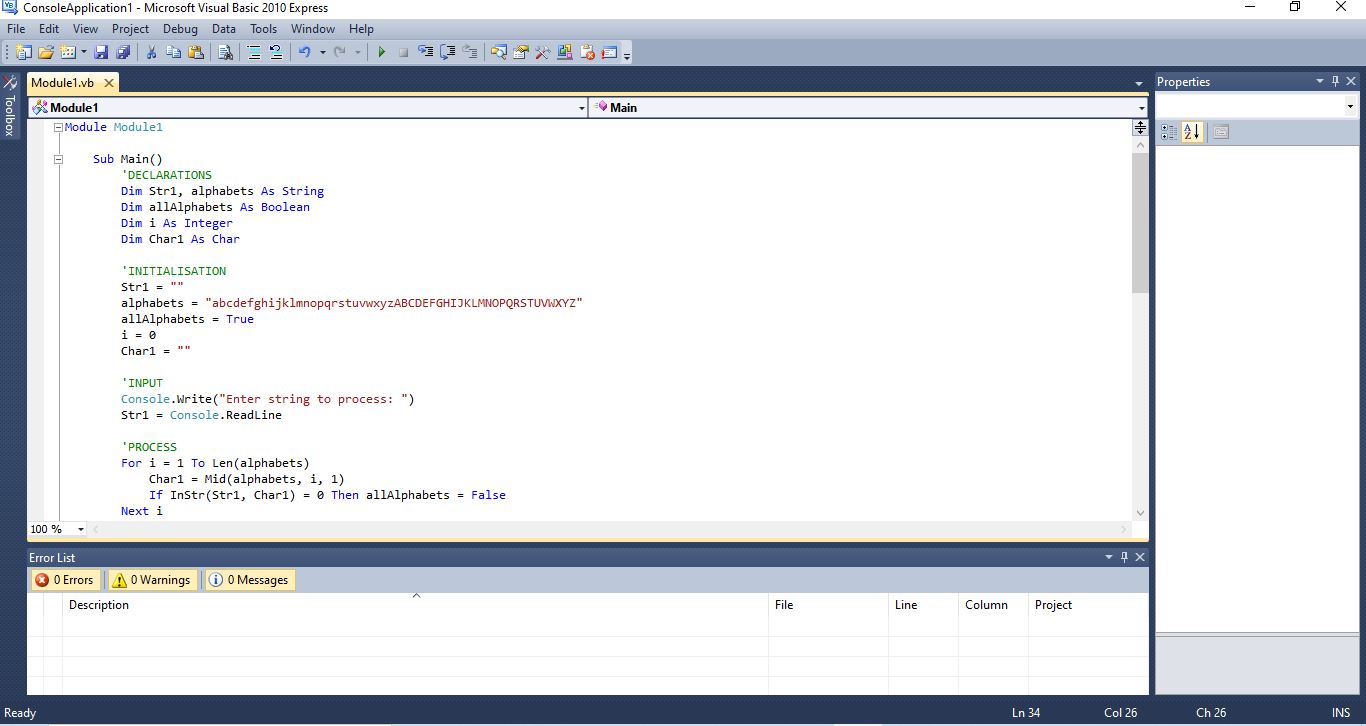
**STRING MANIPULATION ASSIGNMENT**

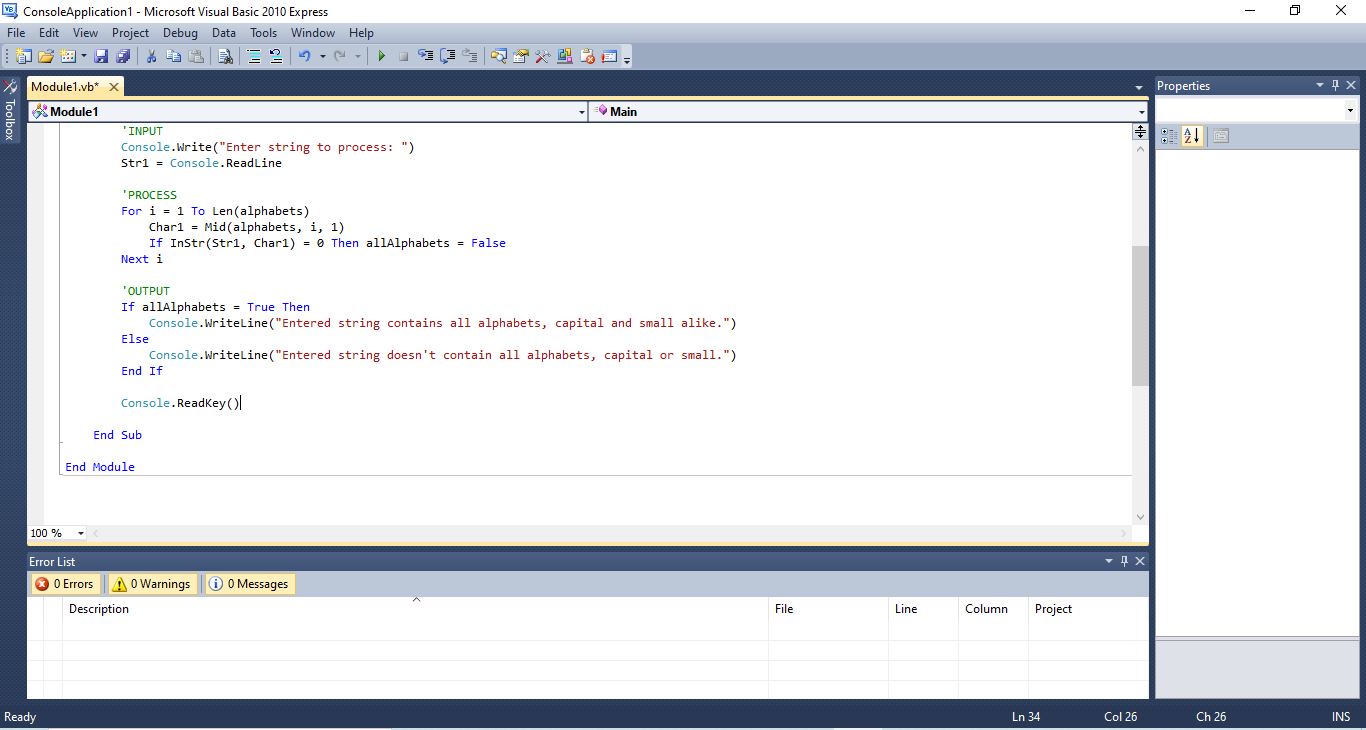
ALINA SIDDIQUI AS-C

1. **Find if the input string has all of the alphabets; capital and small alike.**

**PSEUDOCODE:**

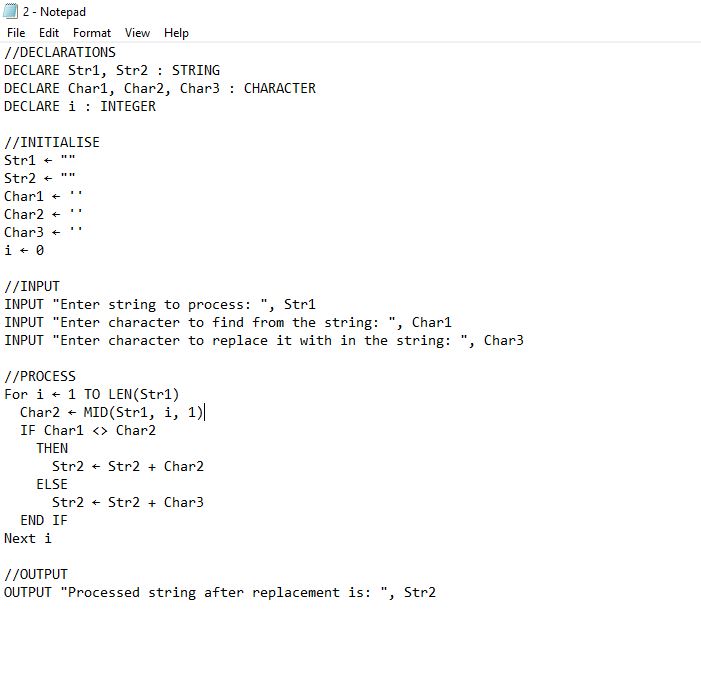
****

**VISUAL BASIC: **

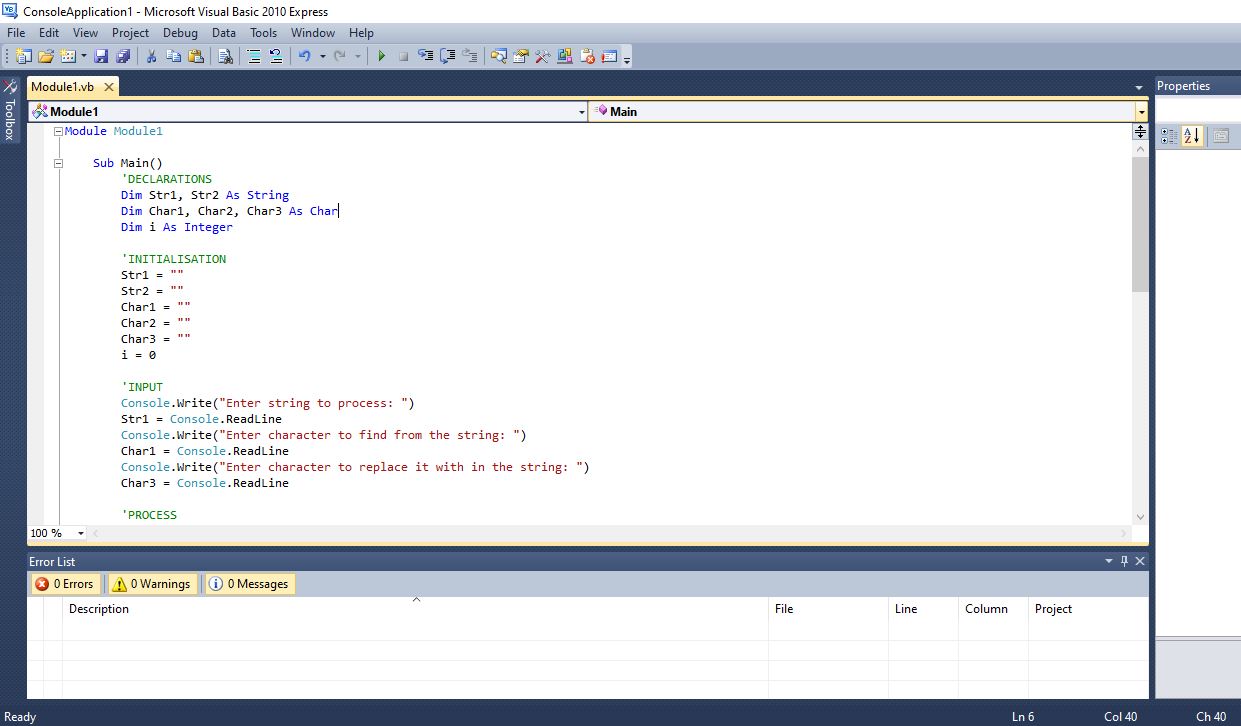
****

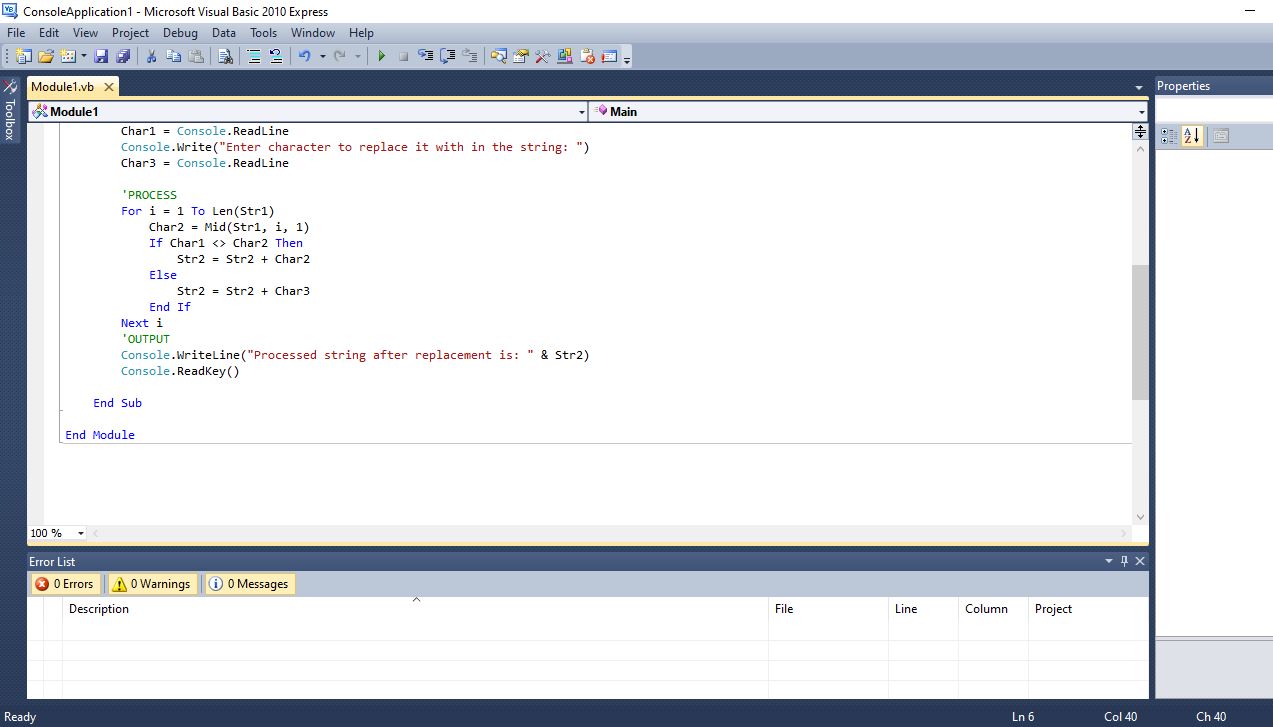
1. **Replace selected character with another in entered string.**

**PSEUDOCODE:**

****

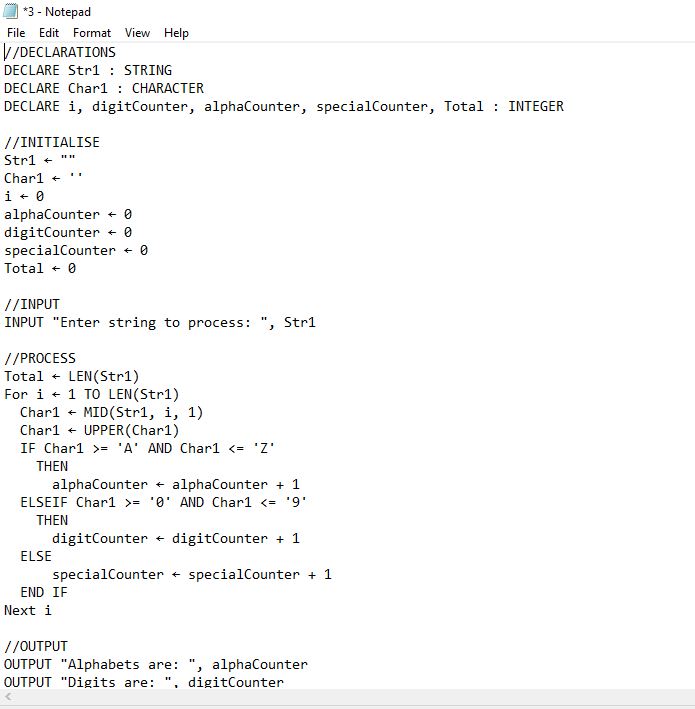
**VISUAL BASIC:**

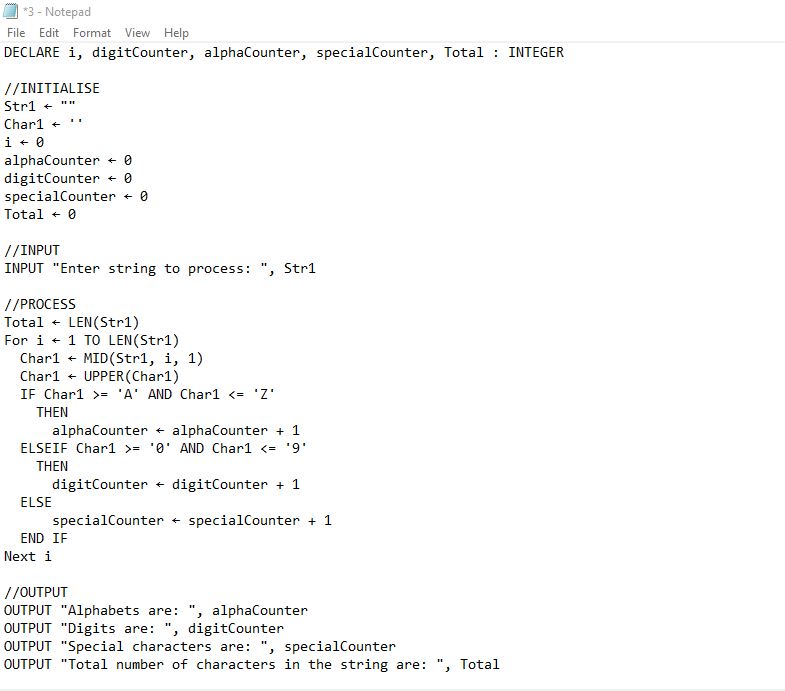
****

****

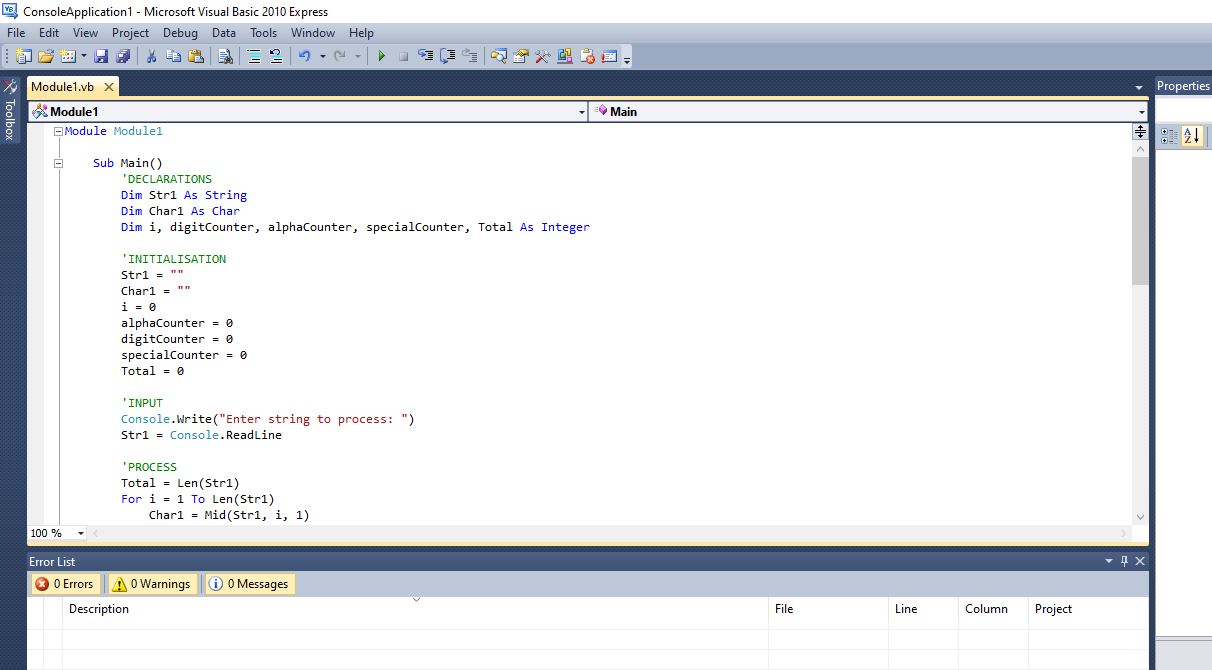
1. **Count and output the NUMBER of an entered character. Output separate counts for alphabets (cap & small together), digits and other characters.**

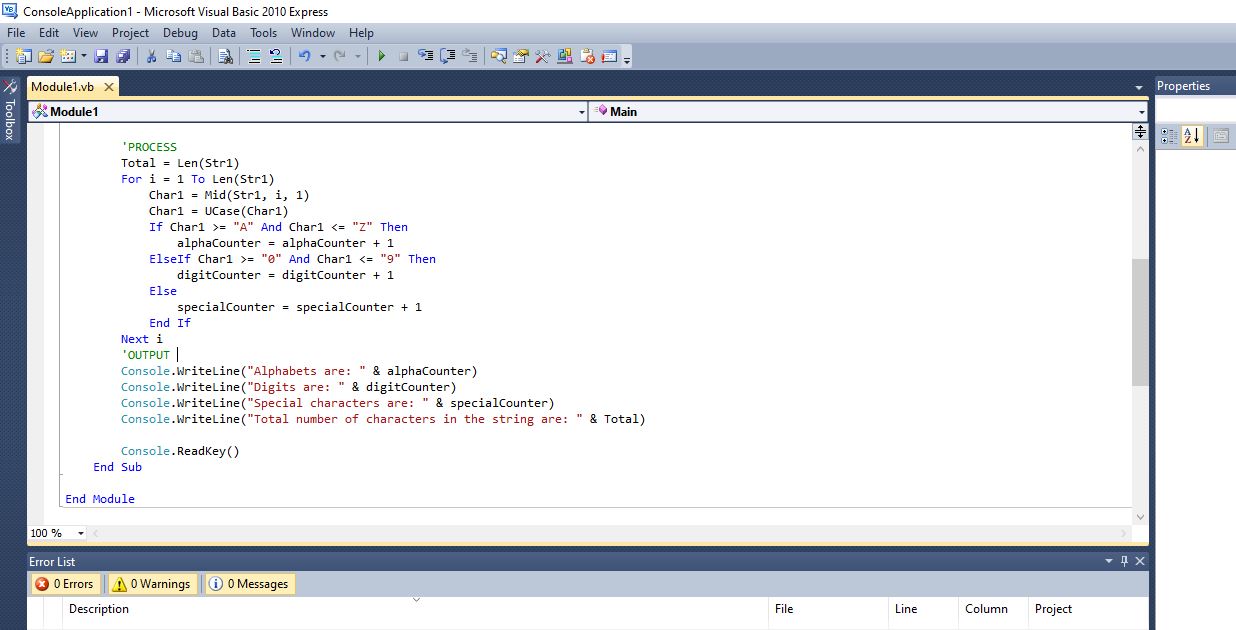
**PSEUDOCODE:**

****

****

**VISUAL BASIC:**





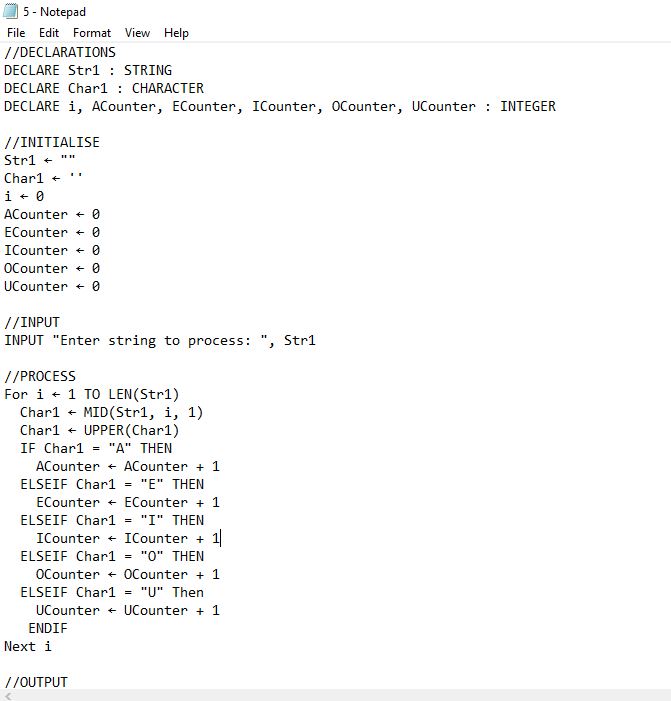
1. **Find the character that appears most number of times in an entered string and output it.**

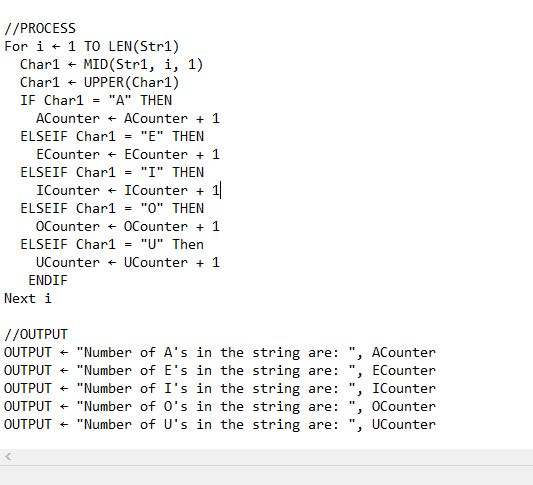
**PSEUDOCODE:**

**VISUAL BASIC:**

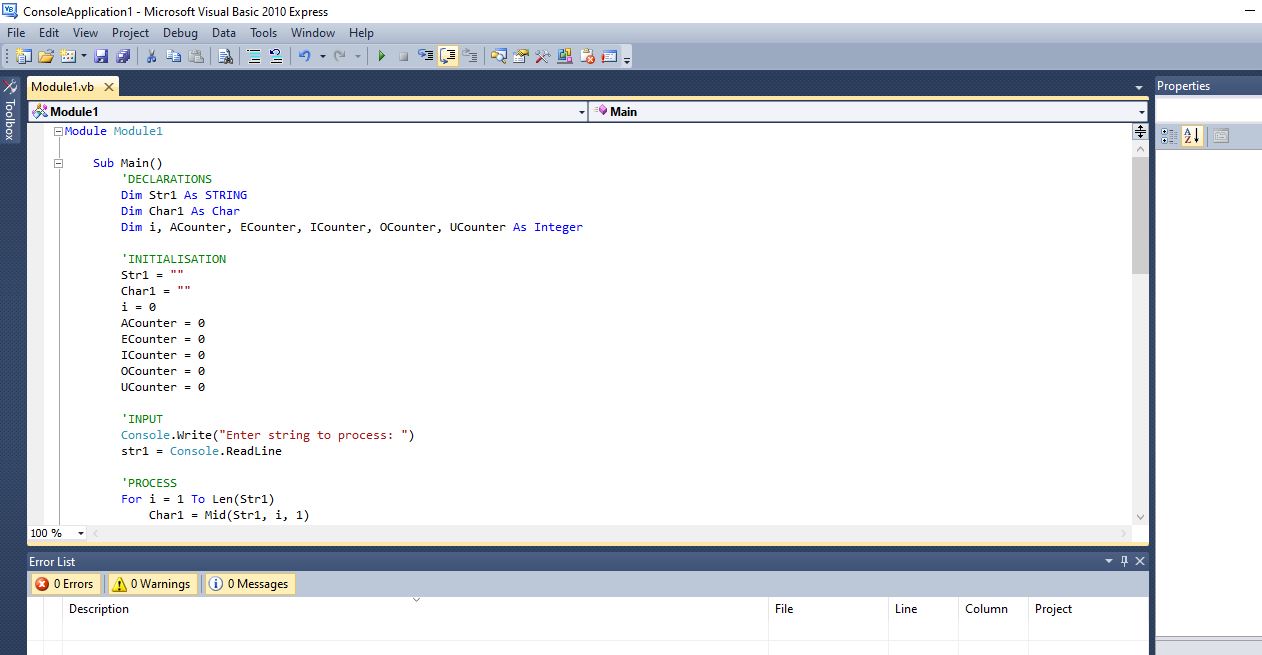
1. **Find the count of vowels characters in an entered string separately.**

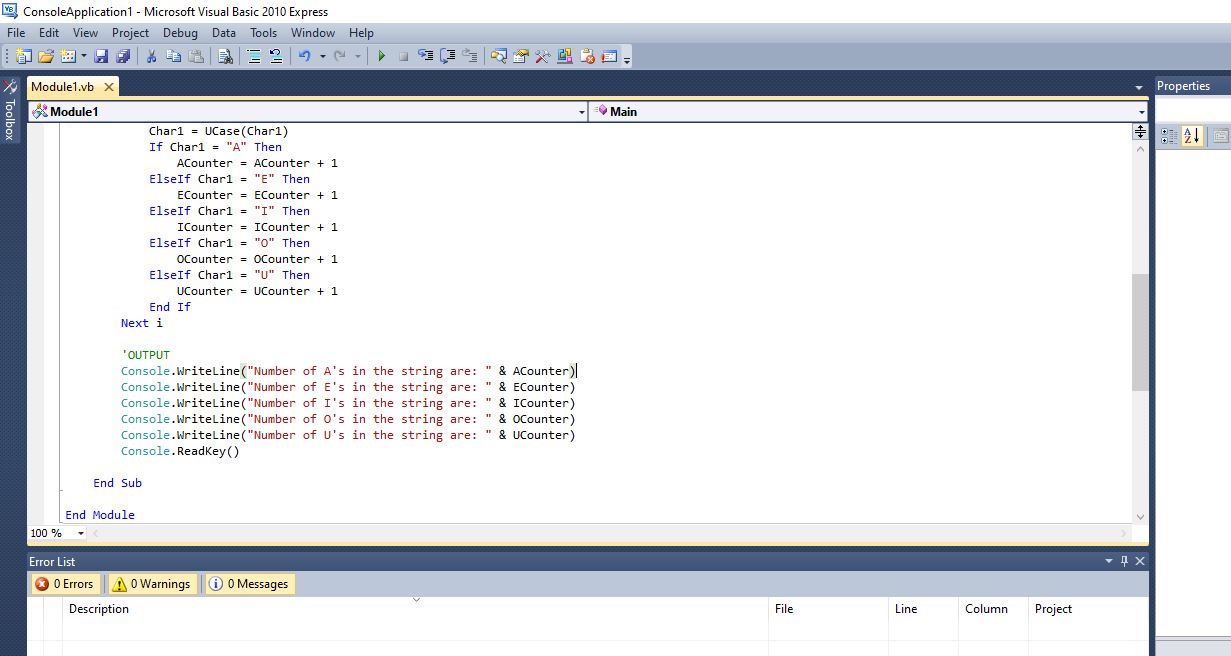
**PSEUDOCODE:**

****

****

**VISUAL BASIC:**

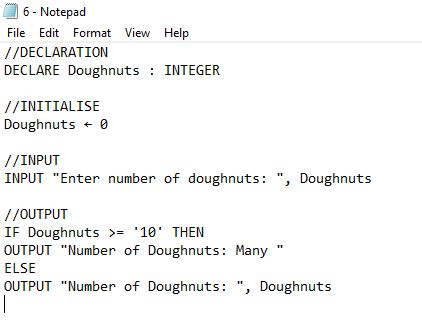
****

****

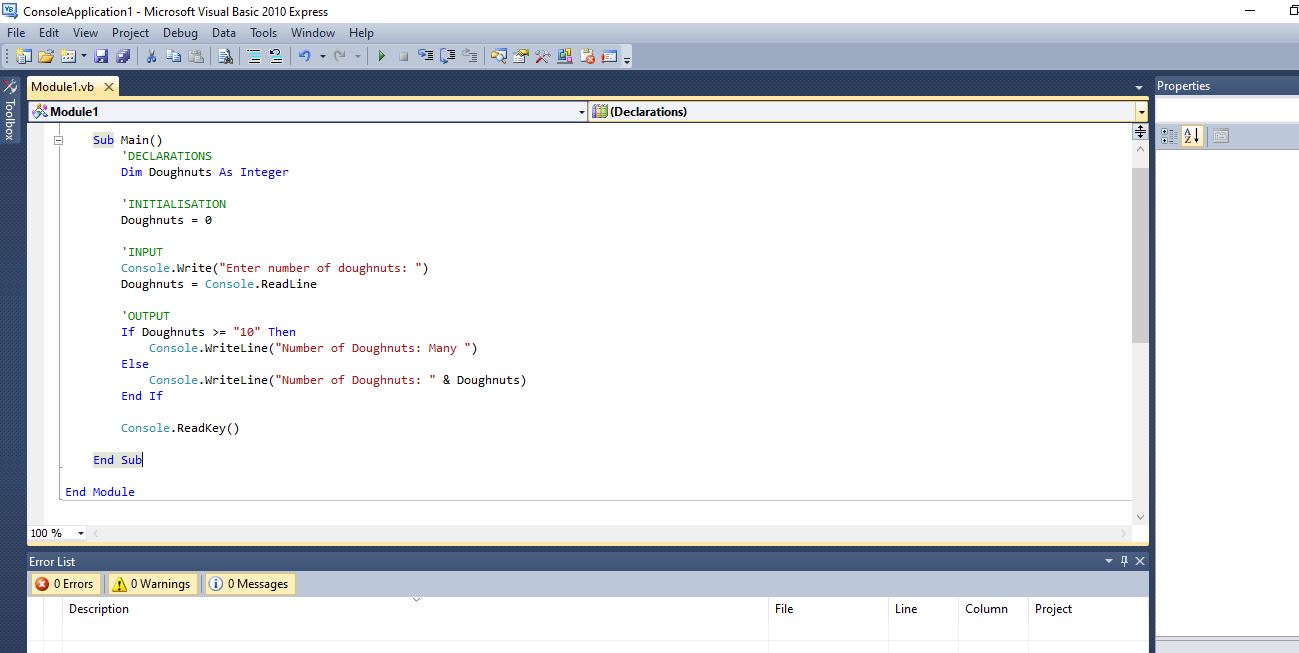
1. **donuts:**

**INPUT an INT count of a number of donuts, OUTPUT a string of the form 'Number of donuts: <count>', where <count> is the number input. However, if the count is 10 or more, then use the word 'many' instead of the actual count. So donuts (5) outputs 'Number of donuts: 5' and donuts (23) outputs 'Number of donuts: many'**

**PSEUDOCODE:**

****

**VISUAL BASIC:**

****

1. **both\_ends:**

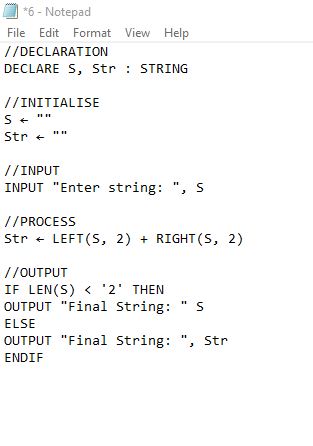
**Input a string s, output a string made of the first 2**

**and the last 2 chars of the original string,**

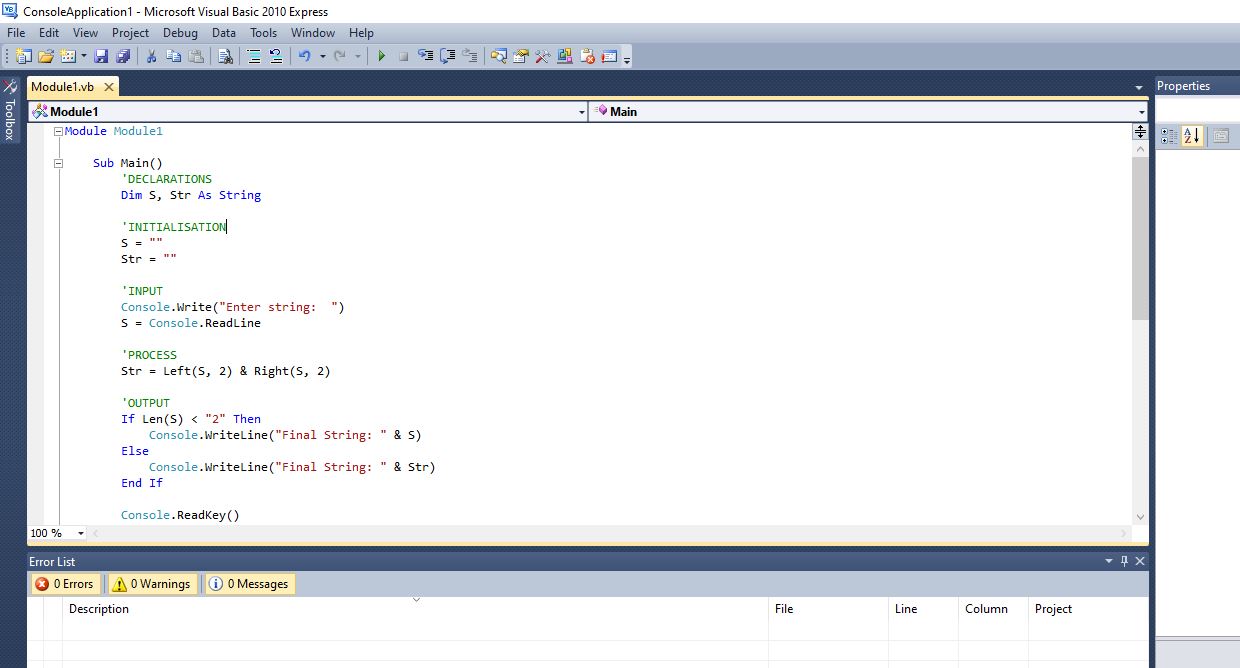
**so 'spring' yields 'spng'. However, if the string length**

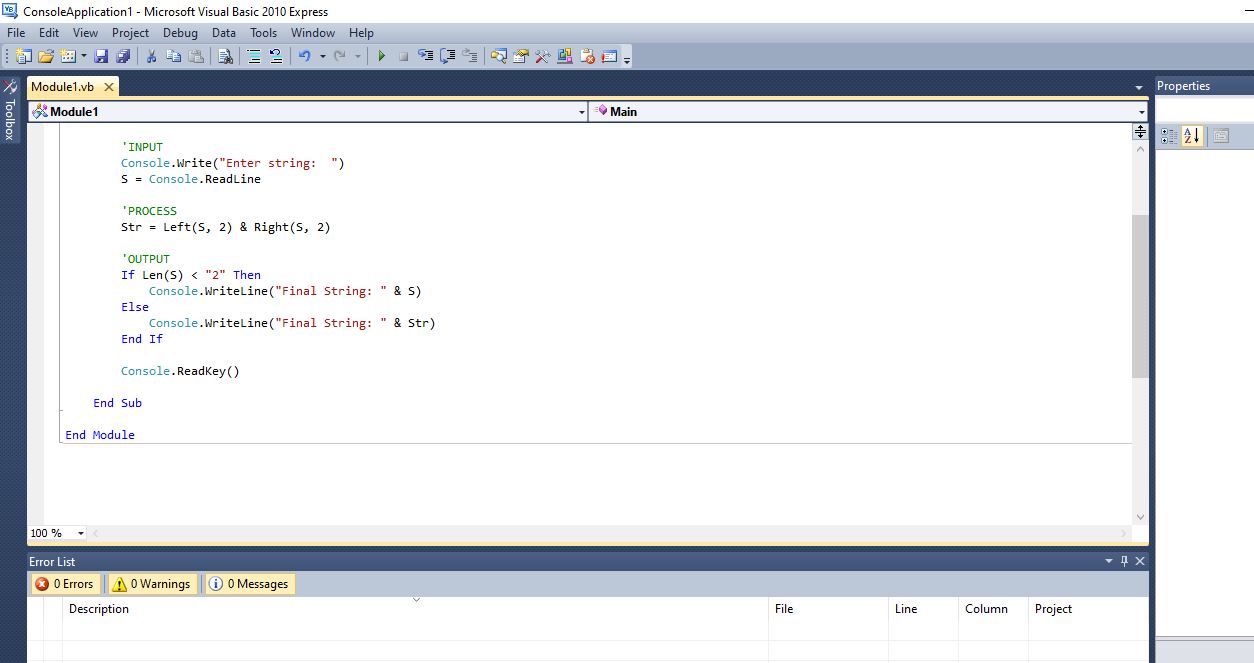
**is less than 2, return string s instead the empty string.**

**PSEUDOCODE:**

****

**VISUAL BASIC:**

****

****

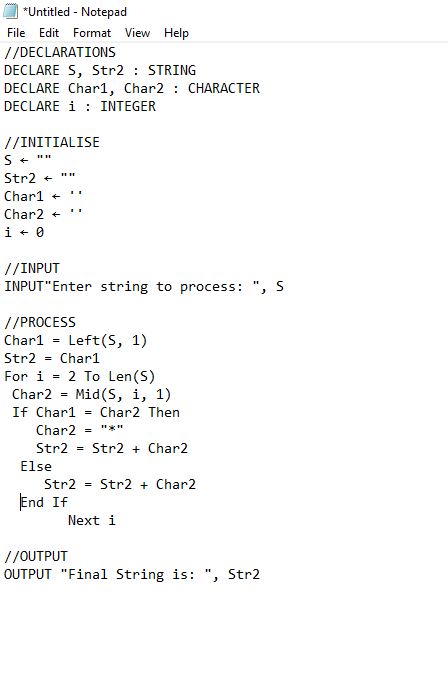
1. **fix\_start: Given a string s, return a string**

**where all occurrences of its first char have**

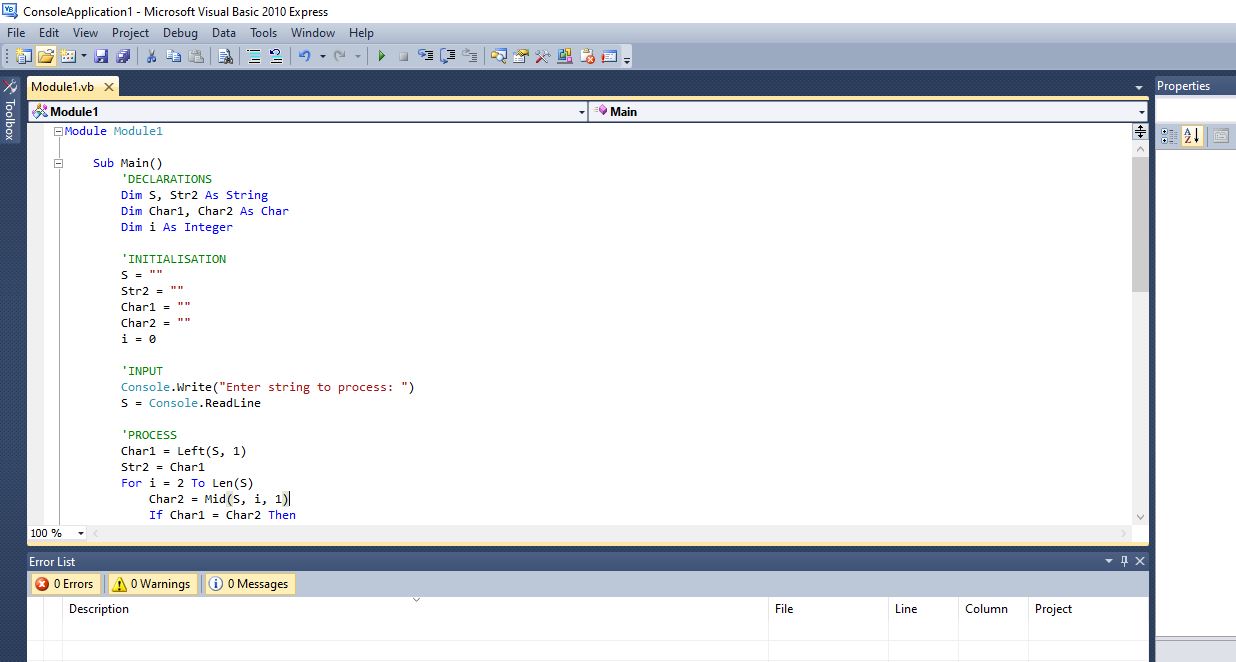
**been changed to '\*', except do not change**

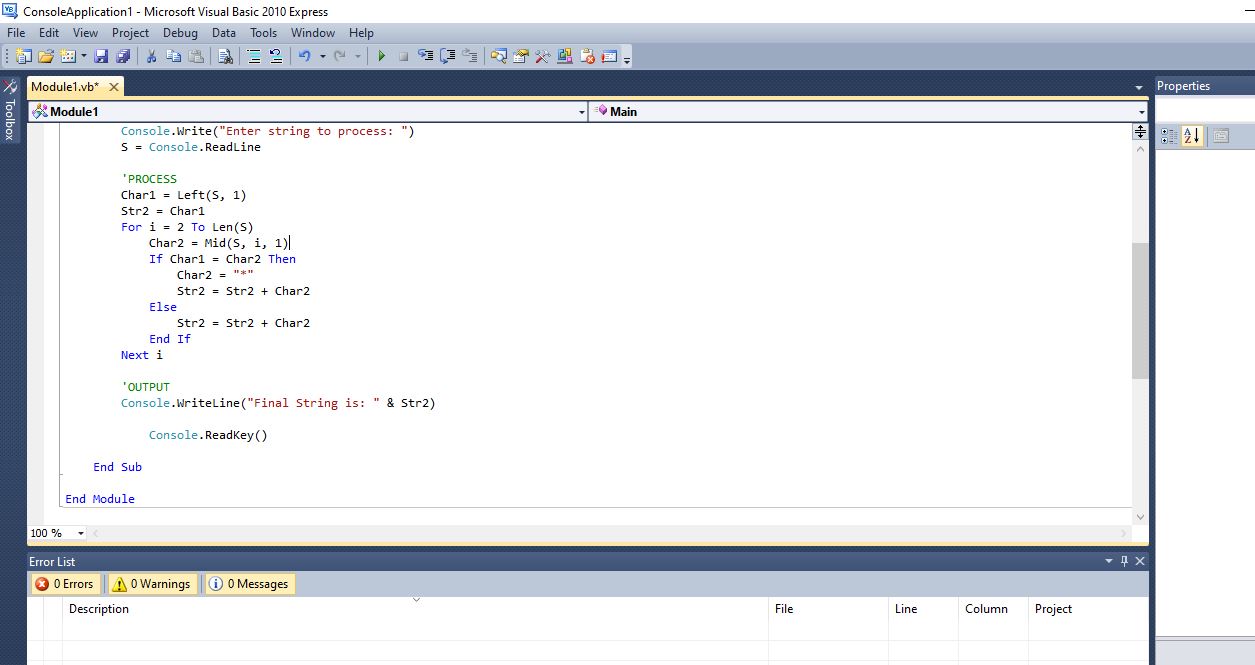
**the first char itself. e.g. 'babble' yields 'ba\*le' or 'alpha' outputs 'alph'. Assume that the string is length 1 or more**

**PSEUDOCODE:**

****

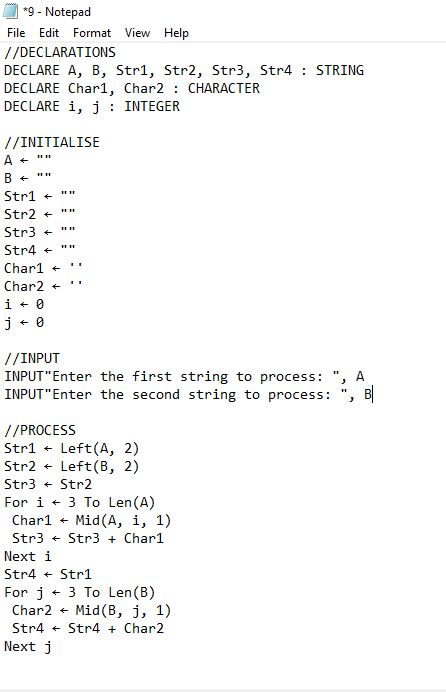
**VISUAL BASIC:**

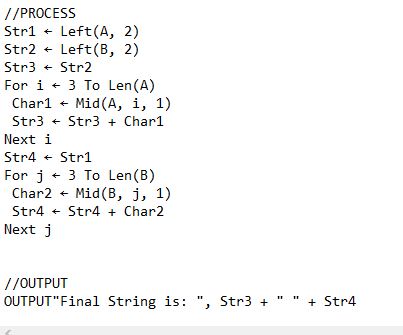
****

****

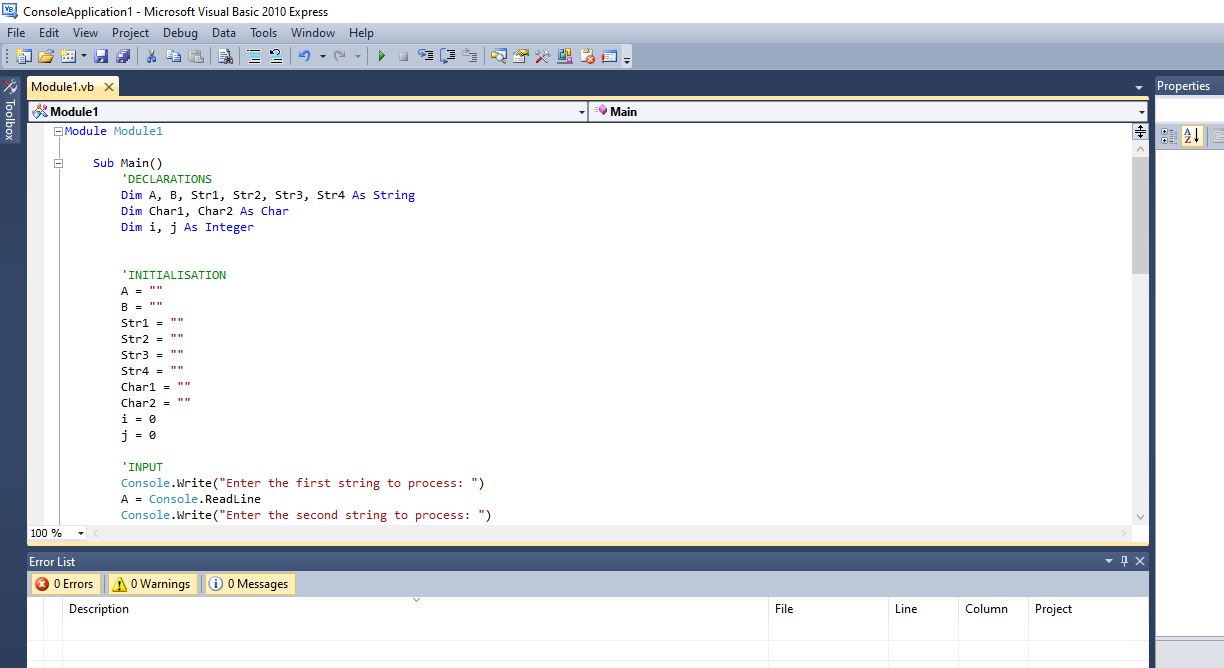
1. **MixUp: Given strings a and b, return a single string with a and b separated by a space '<a> <b>', except swap the first 2 chars of each string. e.g. 'mix', pod' -> 'pox mid', 'dog', 'dinner' -> 'dig donner'. Assume a and b are length 2 or more.**

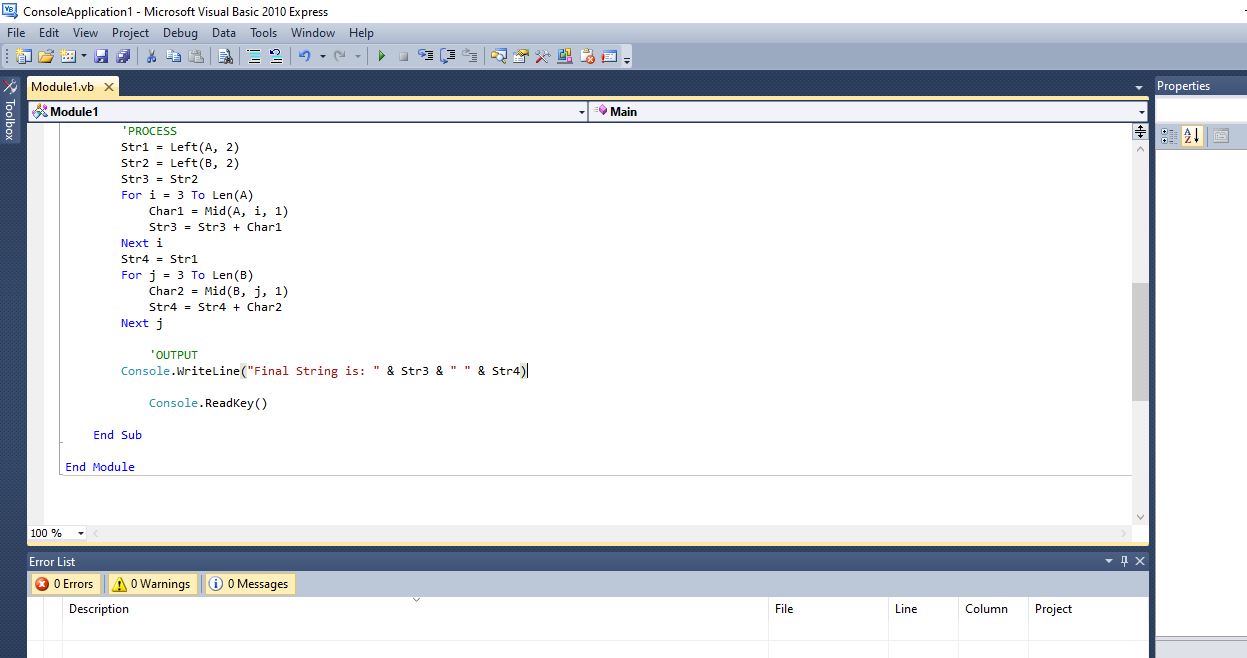
**PSEUDOCODE:**

****

****

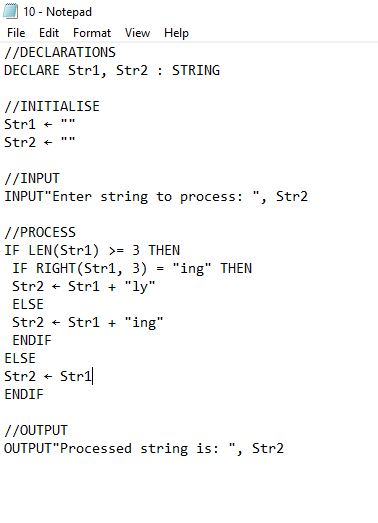
**VISUALBASIC:**

****

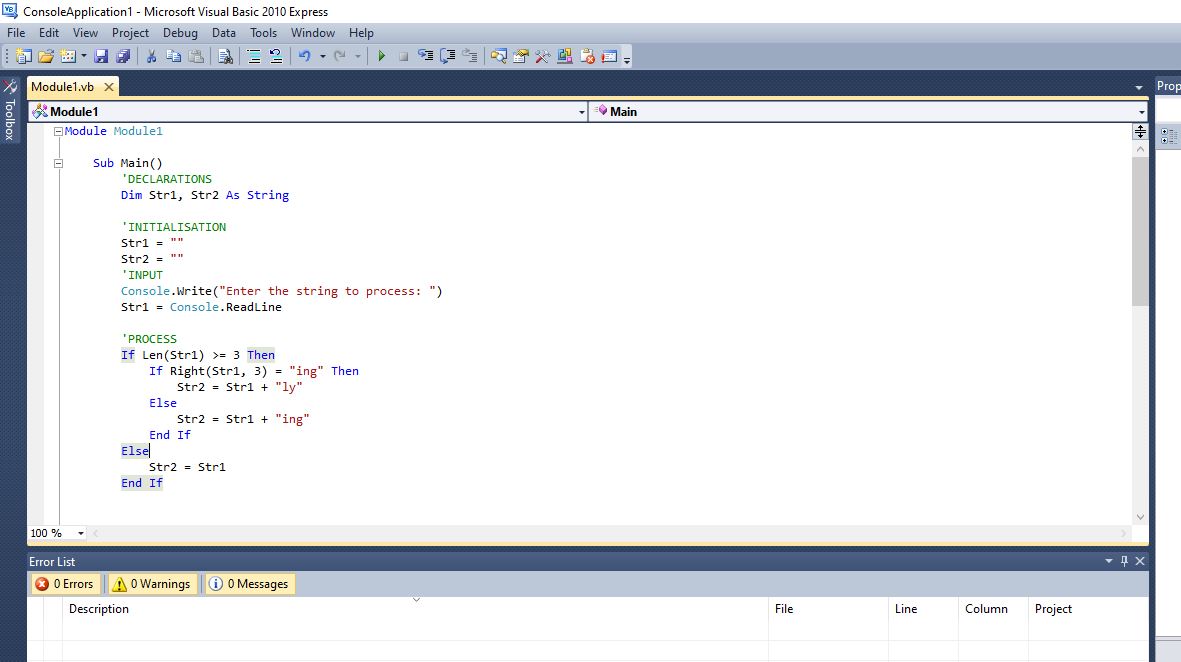
****

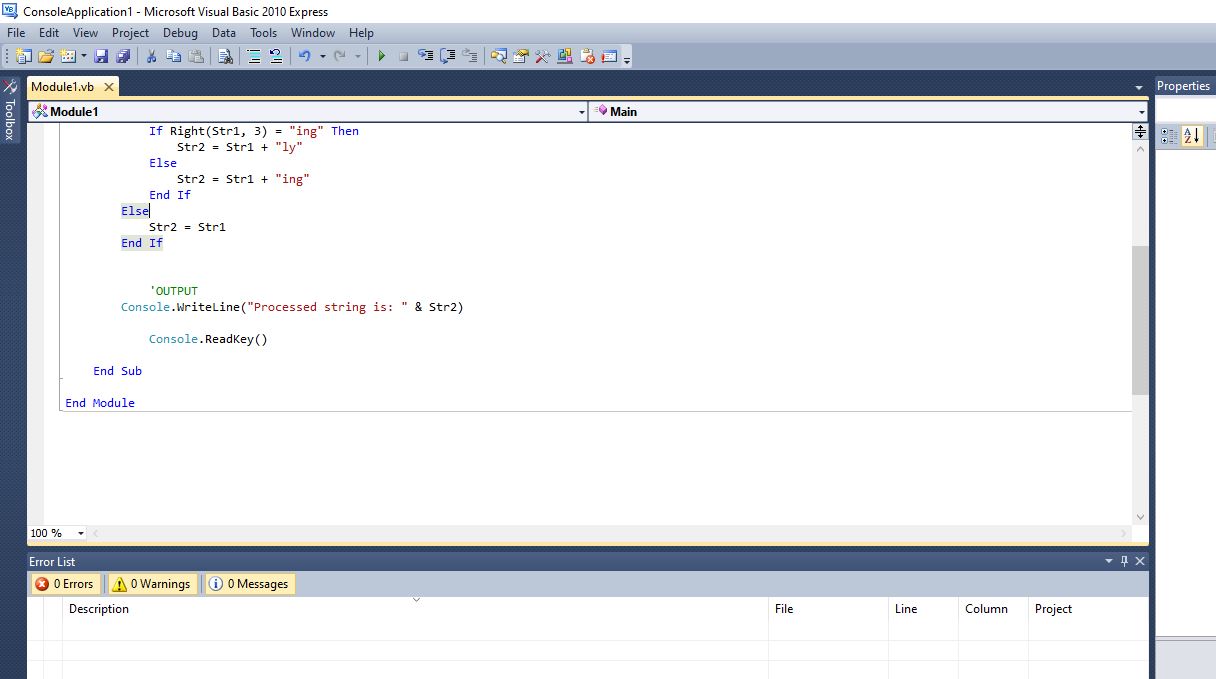
1. **verbing: Given a string, if its length is at least 3, add 'ing' to its end. Unless it already ends in 'ing', which case add 'ly' instead. If the string length is less than 3, leave it unchanged. Return the resulting string. examples: input 'end' --> 'ending', input 'ending' --> 'endingly', input 'go' --> 'go'**

**PSEUDOCODE:**

****

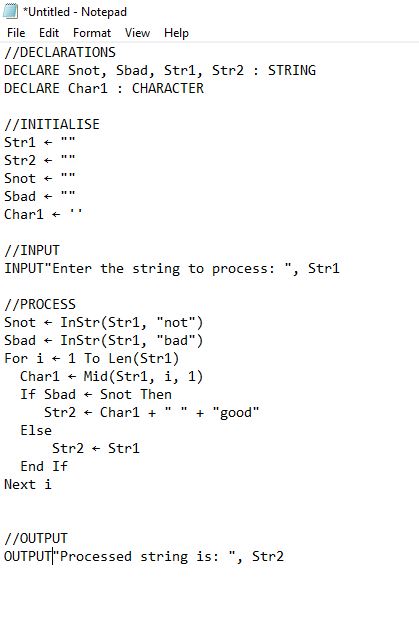
**VISUAL BASIC:**

****

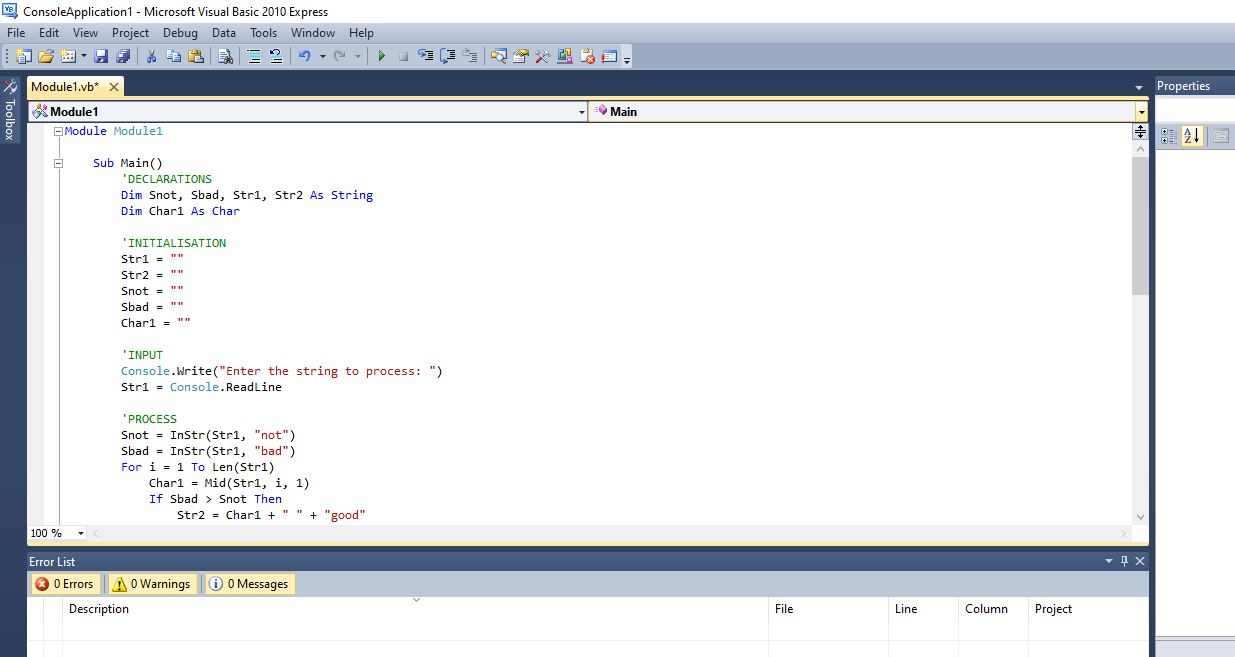
****

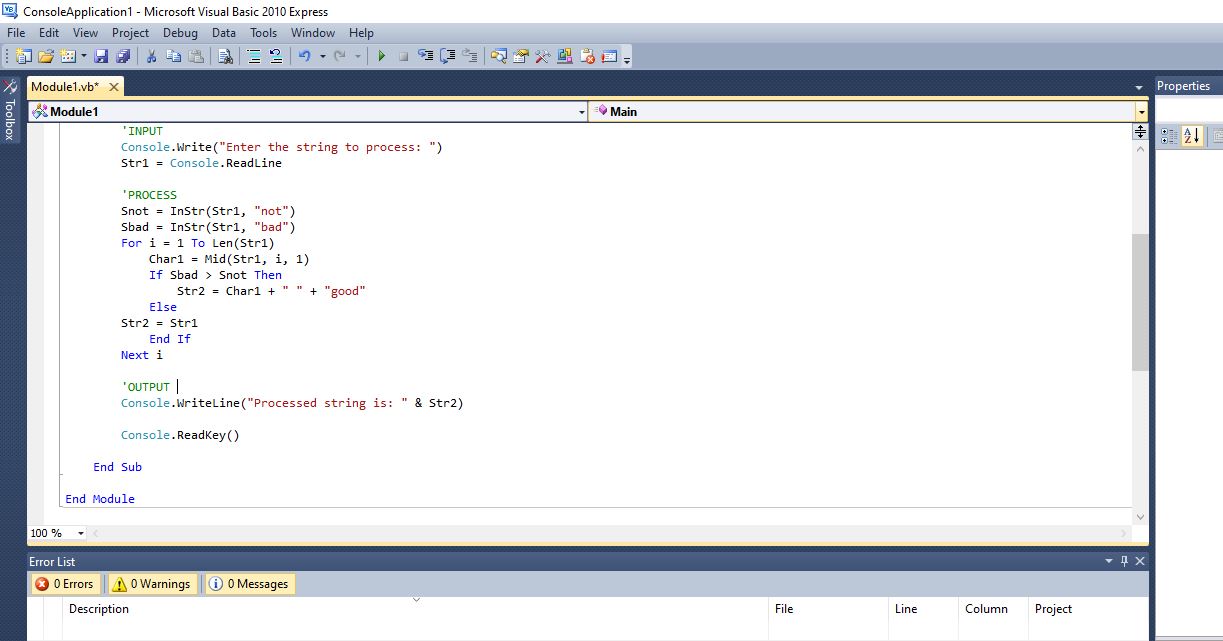
1. **not\_bad: Given a string, find the first appearance of the substring 'not' and 'bad'. If the 'bad' follows the 'not', replace the whole 'not'...'bad' substring with 'good'. Return the resulting string. Input: 'This dinner is not that bad!' Outputs: This dinner is good!**

**PSEUDOCODE:**

****

**VISUAL BASIC:**

****

****