Day 79/180 Recursion in Strings

1: Check Palindrome

Code and logic both are explained in the video.

For this problem: pass the string by reference to the recursive function
To beat the time limit, because if you pass by value, every
time will create a new string copy, that will increase the
the time complexity of your solution.

```
// Pass by Value - create a copy for every call
// Time Complexity for every call: O(n)
// OverAll Time Complexity : O(n^2)

int check(string s, int i, int j)

// Pass by Reference, same string will get passed
// OverAll Time Complexity : O(n)
int check(string &s, int i, int j)
```

2: Lower case to upper case

Code and logic both are explained in the video.

3: Convert String to LowerCase

- Similar to lowercase to upper case, but here character can be already an upper case, so, you have to skip those.

```
// Function to convert a character at a given index to lowercase
void to_lower(string &s, int index)
{
    // Base case: If index is -1, stop recursion
    if(index == -1)
        return;

    // Check if the character at the current index is an uppercase
letter
    if(s[index] >='A' && s[index] <='Z')
    {
        // Convert the uppercase letter to lowercase
        s[index] = 'a' + s[index]-'A';
    }

    // Recursive call to process the next character in the string
    to_lower(s, index-1);
}

// Function to convert the entire string to lowercase
string toLower(string S) {
    // Start the recursion from the last character of the string
    to_lower(S, S.size()-1);

    // Return the modified string in lowercase
    return S;
}
```

4: Reverse a String

Code and logic both are explained in the video.

5: Given a String, count the number of consonants in it.

- Similar to the problem discussed in the video, where we were counting vowels.
- Here, we're increasing count if the current character isn't a vowel.

```cpp
#include <iostream>
using namespace std;

int count(string str, int index)
{
    if(index == -1)
    {
        return 0;
    }

    if(str[index] == 'a' || str[index] == 'e' || str[index] ==
'i' || str[index] == 'o' || str[index] == 'u'  )
    {
        return  count(str,index-1);
    }
    else
    {
        return 1 + count(str,index-1);
    }
}

int main() {
    // Write C++ code here

    string str = "coders";
    cout<<"count of consonants : "<<count(str,5);
    return 0;
}
```