Q. leet code: 344 Reverse String.

Q. code studio: check if the string is a Palindrome

Q. leet code: 125 valid palindrome.

Q. leet code: 186 Reverse words in a string II

Q. GG : maximum occurance character

Q. code studio: Replace Spaces

Q. leetcode: 1910 : Remove All occurances of a substring.

Q. leetcode: 567: permutation in string.

Q. leetcode: 1047: Remove all adjacent duplicates In string.

Q. leetcode: 443: String compression

* char & Strings *

→ Strings: Strings are one dimensional char array.

← char array.

String → | a | b | c | d | e | f | g | h |

→ char array:

701

char ch[10]; | a | b | c | d | - | - | - | - | - | h |

ch → 701

→ input:

char name[20];

I/P cin >> name; Babbar

| B | a | b | b | a | r | \0 | - | - | - | - | - | - | - | - | - | - | - | - | - |

0 1 2 3 4 5 6 19

→ as i/p finish it adds null char '\0' at end

use as a terminator.

→ length: strlen(name);

→ compare: strcmp(s1, s2);

→ copy: strcpy(dest, src);

→ O/P cout << name; → Babbar

Note: cin stops execution in the case of char array when you give space, tab, enter

Q. length of String?

```
for(int i=0; name[i]!='\0'; i++){
    count++;
}
```

Q. Reverse a string?

Algo: find length; s=0; e=length-1;

```
while(s<e){
    swap(ch[s], ch[e]);
    s++;
    e--;
}
```

* String: String stores its data internally in the form of a null-terminated C-string, but in normal usage does not allow you to access the null terminator

→ length():
```
string str;
str.length();
```

→ .push_back('c');

→ .pop_back();

→ .find(part)

→ .erase(s_index, e_index); (exclusive)

* key differences between char array & string:

A string is collection of characters expressed as an individual data type. A character Array is set of char data in sequential order. The char function may retrieve characters in a string at a certain index. The symbols in char array could be retrieved in the same way they are from any other language.

→ cin.getline(str, len); (custom delimeter ?)

Q. leetcode: 443

```
int i=0;
int ansIndex=0;
int n= chars.size();

while(i<n){
    j=i+1;
    while(j<n && chars[i]==chars[j]){
        j++;
    }

    chars[ansIndex++]=chars[i];
    int count=j-i;
    if(count>1){
        string cnt= to_string(count);
        for(char ch:cnt){
            chars[ansIndex++]=ch;
        }
    }
    i=j;
}
return ansIndex;
```