Today's Agenda :-

Intro
Flip
Sort ch[]
Reverse String
Longest Palindromic Substring.

Quote :-

No One gets Rewarded for
'I had the same idea'

String :- → array of characters
→ Bunch of characters → a c b / abc
→ Sequence of characters

* order is also important.

characters. → unicode in java. → 2 bytes.
Ascii → 1 byte.

'A' → 65	'a' → 97	'0' → 48
B → 66	'b' → 98	'1' → 49
C → 67	'c' → 99	'2' → 50
D → 68	'd' → 100	.
⋮		⋮
Z → 90	'z' → 122	'9' → 57
		'10' → ? It's not a char.

+32 →, ← −32

char ch = '9' → ch = ch + 8
1 byte, ↳ Ascii = 57

print(ch).
A.

7	6	5	4	3	2	1	0
0	0	1	1	1	0	0	1

String → Array of characters.

String s = "abda",
Print (s[0]) ⇒ a.

chars s[4] = 'abcd'
print (s[0]) → a.

Ques) Given a char [], Toggle every character.

Note :- Input will contain only small and capital letters.

ch[] → Ana Conda

Toggle

ch[] → aNAcONDA

```
Toggle ( char s[]) {    T.C → O(n), S.C → O(1).

    int n = s.length;
    for (i=0; i<n; i++) {
        if (s[i] >= 'A' && s[i] <= 'Z') {
            s[i] = s[i] + 32
        }
        else {
            s[i] = s[i] - 32
        }
    }
    return s
}
```

s[i] = s[i] ^ 32

or

s[i] = s[i] ^ (1<<5)

'A' ⇒ 65

7	6	5	4	3	2	1	0
0	1	0	0	0	0	0	1

'a' → 97

7	6	5	4	3	2	1	0
0	1	1	0	0	0	0	1

'B' ⇒ 66

0	1	0	0	0	0	1	0

'b' → 98

0	1	1	0	0	0	1	0

20) Given a char ch[], which contains only lowercase alphabets, sort given ch[], in alphabetical order, S.C. O(1).

Ex:- s = d a b a c d b

After ↓ Sort

s = a a b b c d d

Constraints :- $1 <= n <= 10^5$

1) using inbuilt sort
T.C → O(n log n).
S.C → O(n).

2) use Bubble sort
T.C → O(n^2).

idea 3:- Counting all characters :-

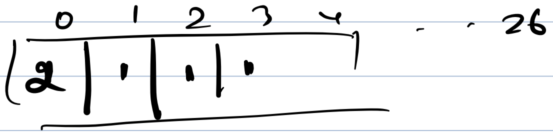
s = d a b a c d b ⟶ int cnt[26] = {0}

'a' → 2
'b' → 2
'c' → 1
'd' → 2

'a' → -'a' cnt [0]
b → -'a' cnt[1]
c → -'a'
z → -'a' cnt [25]

s = a a b b c d d

a b c d a



```
- Sort String ( char s[]) {

        int   n = s.length;
        int   c[26] = {0}.
        for (i=0; i<n; i++) {
                idx = s[i] - 'a'
                c[idx] ++
        }

        int k=0;
          for(i=0; i<26; i++) {
               char ch =   'a' + i
               for (j = 1; j <= c[i] ; j++){
                    s[k] = ch;  k++
               }
          }
}
```

curr ch

'a' - 'a' → 0
'b' - 'a' → 1
'c' - 'a' → 2

T.C → O(n)
S.C → O(1)

$\} \rightarrow 26 * n$
$\rightarrow n$

s = " ~~ddd~~ ~~e~~cc b aaa'

aaa b

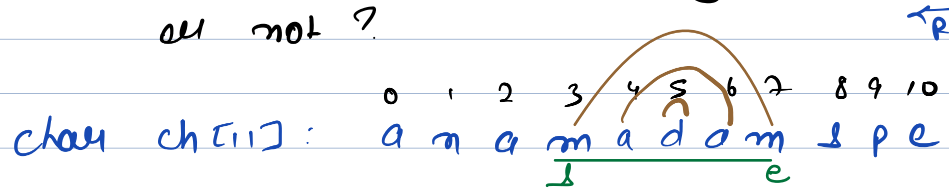
	0	1	2	3	4	5	.	-	-
c =	3	1	3	3					

i	j	
0	1 to 3	3
1	1 to 1	1
2	1 to 3	3
3	1 to 3	3
		10

Substring :-
- ↳ continuous part of a string
- ↳ full length ✓
- ↳ single char ✓

Ques) check if given substring is Palindrome or not ?

$L \rightarrow R$ / $R \leftarrow L$



NITIN
MADAM
MALAYALAM
KANAK
NAYAN
DID

```
bool isPalin(char ch[], int s, int e)
    while ( s<e) {
        if (ch[s]!= ch[e]) {
            return false
        }
        s = s+1, e = e-1
    }
    return True;
}
```

T.C → O(N)
S.C → O(1).

↪ Break.

10:03pm to 10:13 pm

Ques) Given a string, calculate the length of longest Palindromic substring.

Ex1:-)

```
0 1 2 3 4 5
a b a c a b   -->  n(n+1)/2
  ↓
Ans=5.
```

Ex2:-

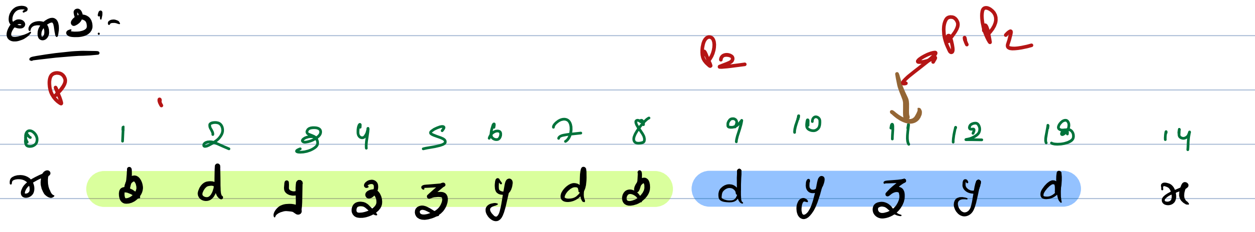
```
0 1 2 3 4
a b c d e  -->  Ans=1.
```

Idea:- for every substring, check if palindrome or not.

```
int longestPalin(char s[]) {        T.C -> O(N^3)
                                    S.C -> O(1).
    int n = s.length;
    int ans = 0;
    for (i=0; i<n; i++) {
        for (j=i; j<n; j++) {
            //substring s[i,j]
            if (isPalin(s, i, j)) {
                ans = max(ans, j-i+1)
            }
        }
    }

    return ans;
}
```

Ex:-



idea:-

$(P_1 \quad P_2)$

$[P_1+1, P_2-1] \rightarrow P_2-P_1-1$

Take every character as centre & Expand on centre and get max palindromic substring length.

$T.C \rightarrow O(N) * n \Rightarrow O(n^2)$.

Ignoring even length Palindrome.

→ Take every adjacent characters as centre & expand on centre & get max palindromic substring len.

$T.C \rightarrow O(N) * O(N) \Rightarrow O(N^2)$.

```
int expand ( char s[], int p1 , int p2 ){
    while ( p1>=0 && p2<n && s[p1] == s[p2] ) {
            p1 --; p2++
      }
    return p2-p1-1;
}

int longPalin ( char s[]) {
      int n = s.length;
      int ans = 0/1;
      for ( i=0; i<n; i++ ) {        -> odd cases.
              // centre  s[i];
               p1 = i, p2 = i
               ans = max ( ans, expand ( s, p1, p2 ))
      }

      for ( i=0; i<n-1 ; i++) {      -> Even cases.
              // Centre  s[i], s[i+1]
              p1 = i, p2 = i+1
              ans = max ( ans, expand ( s, p1, p2 ))
      }

      return ans ;
}
```

T.C -> $O(n^2)$.

$$\begin{tikzcd}[column sep=small]
 & & \text{Longest Palindromic Substring.} \arrow[dll] \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & \\
O(N^3) & O(N^2) & O(N^2) & O(N \log N) & O(N)\ (\text{Manacher's})
\end{tikzcd}$$

BF | Expand in centre | using dp | B.S + Rabin karp. | ↳ optimal.

$$\begin{tikzcd}[column sep=small]
 & & (0-5) \arrow[dl] \arrow[dr] & & \\
 & (0-4) \arrow[dl] \arrow[d] & & (1-5) \arrow[d] \arrow[dr] & \\
(0-3) & (1-4) & & (1-4) & (2-5)
\end{tikzcd}$$

visited	T	T	T	T	T
	0	1	2	3	4

1,	2,	15	4	3
0	1	2	3	4

O(n).

$$\begin{tikzcd}
 & 1 \arrow[dr] & & \\
0 \arrow[ur] & & 2 \arrow[r] & 0
\end{tikzcd}$$

$0 \rightarrow 1 \rightarrow 2 \rightarrow 15$.

$3 \rightarrow 4 \rightarrow 3$