Strings Introduction

- Intro
- Flip
- Sort the ch[]
- If substring is Palindrome
- Longest palindromic substring

Strings :

→ Array of characters
→ Sequence of chars
→ Set of chars
→ Bunch / Group of chars.

{a b d} order
{a d b} matters

'A' → 65 → Binary representation

ASCII

5th bit

'A'	→	65
'B'	→	66
'C'	→	67
'D'		68
⋮		⋮
'Z'		90

$\xleftrightarrow{32}$

5th bit is set

'a'	→	97
'b'	→	98
'c'		99
'd'		100
⋮		⋮
'z'		122

'0'	→	48
'1'	→	49
⋮		⋮
'9'		57

B/w Capital & small char, ASCII value diff is 32.

Strings

- → String { we can't chang them } In Java ↓ immutable
- → char s[] { we can change a particular index }

```
String s = "Hello"
   s.charAt(0)        ✗

char s[5] = {'H','E','l','l','o'}
   s[0] = 'A'         ✓
```

Q. Given a char[], Toggle every char

↓

Capital → small

Small → capital

: Only lower/upper case characters are present.

s[] = AnaConDa

↓

ans = aNAcONdA

```
char[]  Toggle ( char s[])
{
        int n = s.length()
        for(i=0 ; i<N; i++)
        {
            // s[i]    If it is uppercase → lower
                       If it is lower → upper

            If( s[i] >= 65 && s[i] <= 90)
            {
                 s[i] = s[i] + 32
            }
            else
            {
                 s[i] = s[i] - 32
            }
        }
        return s
}
```

TC : O(N)
SC : O(1)

TODO ← solve it without if-else

Both works.

If (s[i] ≥ 'A' && s[i] ≤ 'Z')

Q. Given a char ch[], only consists lower-case alphabets ('a'-'z'). Sort the ch[].

Idea1   Arrays.sort   TC : O(NlogN)

idea2

a a b a b b a a b ⇒ a a a a a b b b b

d a b a c d b → a a b b c d d

Count occurences of each character.

```
int Count[26] = {0}

count[0]  → a
count[1]  → b
count[2]  → c
   .
   .
   .
count[25] → z
```

```
int count[26] = {0}
for( i=0; i<N; i++ )
{
    // S[i]
    count[ s[i] - 'a' ]++
}
```

$\boxed{N}$

ch	count[ch-'a']++
'a'	count[0]++
'b'	count[1]++
'c'	count[2]++
⋮	⋮
	count[25]++

```
char s[]   ← same input
int ind = 0
for( i=0 ; i<26 ; i++)
{
    // add count[i] times i th char in the answer
    ch = 'a' + i
    for( j=1 ; j<= count[i] ; j++)
    {
        s[ind] = ch
        ind++
    }
}
return s
```

count[0] + count[1] + count[2] + ... count[25]

$= \boxed{N}$

a a a a a a b b b b c c c c c d d d d e e e . . . . z z z

$\xleftrightarrow{\quad}$ count[0] $\xleftrightarrow{\quad}$ count[1] $\xleftrightarrow{\quad}$ count[2] $\xleftrightarrow{\quad}$ count[3] - - — .

TC: O(N)

SC: O(26) ⇒ O(1)

a a a a a a b b b b c c c c d d d d

a b c d a b b c d b a d d d

count[0] = ~~0~~ ~~1~~ ~~2~~ 3 = 3

count[1] = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4 = 4

count[2] = ~~0~~ ~~1~~ 2 = 2

count[3] = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5 = 5

```
for( i=0 ; i<26 ; i++)
{

}
```

a a a b b b b c c d d d d d

↑
ind

Break

10:24

10:34

Q. Given a char[], i and j. Check if substring (i-j) is palindrome or not.

s =

0	1	2	3	4	5	6	7	8	9	10
a	n	a	m	a	d	a	m	s	p	e

$i = 3$ $j = 7$

ans = True

s =

0	1	2	3	4	5	6	7	8	9	10
a	n	a	m	a	d	a	m	s	p	e

$i = 3$ $j = 8$

ans = False.

s =

0	1	2	3	4	5	6	7	8	9	10
a	n	a	m	a	d	a	m	s	p	e

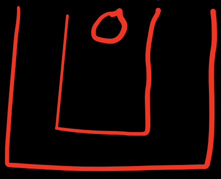
$i = 2$ $j = 4$

ans = True.

$i = 3,$ $j = 7$

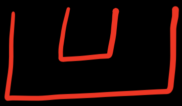
s =

0	1	2	3	4	5	6	7	8	9	10
a	n	a	m	a	d	a	m	s	p	e



$i = 3$	$j = 7$
$i = 4$	$j = 6$
$i = 5$	$j = 5$ ✗

a b b a



$i = 0$	$j = 3$
$i = 1$	$j = 2$
$i = 2$	$j = 1$ ✗

```
bool checkpalindrome ( S[], int i, int j)
{
    while ( i < j)
    {
        If ( S[i] != S[j] )    return False
        i++
        j--
    }
    return True
}
```

TC : O(N)
SC : O(1)

Q. Find len of longest Palindromic substring.

Idea1 : Check every substring.
If Palindromic → update max

$TC : O(N^2 . N) : O(N^3)$

Idea2 :

```
 0  1  2  3  4  5  6  7  8  9  10  11  12  13  14
 a  b  d  y  z  z  y  d  b  d  y   z   y   d   x
 ↑                       ↑  ↑                  ↑
                         i                     j
```

$j - i + 1 - 2$

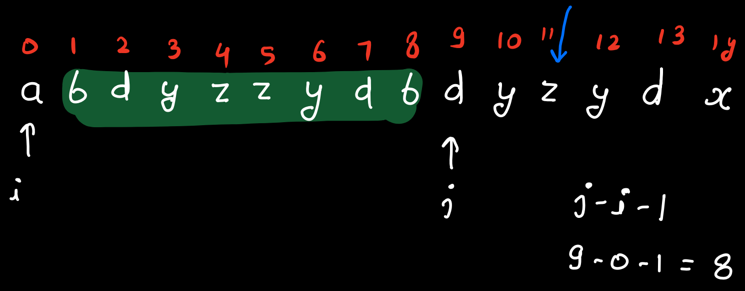
$j - i - 1$

$14 - 8 - 1 = 5$

1. Take each char as center, find len palindromic substring.

2. Take each i & i+1 as centre, find long. Palin. substring.

max



```
int extend ( i, j , s[] )
{
    while ( i ≥ 0 && j < N && s[i] == s[j] )      O(N)
    {  i--, j++
    return j - i - 1
```

```
int lonpalsubs ( s[] )
{                                                  TC: O(N^2)
    int N = s.length                               SC: O(1)
    ans = 0
    for( i = 0; i < N; i++ )
    {  // i is centre.
       int tmpans = extend (i, i, s )               odd length
       ans = max (ans, tmpans)
    for( i = 0 ; i < N-1 ; i++ )
    {  // i & i+1 is centre
       int tmpans = extend (i, i+1, s)              even length
       ans = max (ans, tmpans)
```

return ans

4th Dec (Sunday) 9 pm

Optional Sorting - Comparator

a x y z c a b a c p q r ✗

$$\begin{array}{ccccccccccccc} & & & & & & & & \downarrow & & & & \\ x & y & z & a & a & b & a & a & b & a & a & p & q \\ & & & & & & \uparrow & \uparrow & & & & & \end{array}$$