1. String Basics
2. Toggle each char
3. Check substr palindrome
4. Longest Palindromic Substring

Module → 18 Sept

String → array of chars
→ sequence of chars

"Hello World"   "Apurva"
"Scala"

Characters → single symbol that represents
↑ group of chars
Text

a letter ('a'-'z', 'A'-'Z')
a digit (0, 1, 2, ...9)
symbol ('@', '!' ...)

'@'

char → int → binary
0 and 1

char mapped to ASCII value

ASCII
'A' → 65
'B' → 66
'C' → 67
'D' → 68
⋮
'X' → 88
'Y' → 89
'Z' → 90

'A' → 'Z'
65 - 90

'a' → 'z'
97 - 122

'a' → 97
'b' → 98
'c' → 99
'd' → 100
⋮
'x' → 120
'y' → 121
'z' → 122

		ASCII
'0'	→	48
'1'	→	49
'2'	→	50
'3'	→	51
'4'	→	52
'5'	→	53
'6'	→	54
'7'	→	55
'8'	→	56
'9'	→	57
10	→	X Not a single chars

$$\begin{tikzcd} \text{"1} \arrow[d] & \text{0"} \arrow[d] \\ 49 & 48 \end{tikzcd}$$

No. of chars = 256

Some operations on characters

char ch = 'b' / 'c'

① char ch = 65 → char ch = (char) 65

print (ch) → 'A'

Try char ch = 10489

② char ch = 'a' + 1 $\boxed{98}$

print (ch) → 'b'

$$\begin{tikzcd} \text{char} \arrow[d] & \text{int} \arrow[d, no head] \\ 97 & + \; 1 \end{tikzcd}$$

= 98

```
char  ch = (char) 'a' + 1
```

③
```
int  x = 'a'                x = 97
print  (x)  →  97
```

char + int

↓

$\underbrace{\text{ASCII} \quad + \text{int}}_{\text{int}}$

1. Given a string consisting of alphabets (either uppercase / lowercase). Toggle case of each character and print it.

Uppercase → lowercase
Lowercase → uppercase

s = "Hello"
↓
print → hELLO

" aDgbHJe"
↓
AdGBhjE

		ASCII				
'A'	→	65	$\xrightarrow{+32}$ $\xleftarrow[-32]{}$	'a'	→	97
'B'	→	66	$\xrightarrow{+32}$ $\xleftarrow[-32]{}$	'b'	→	98
'C'	→	67	$\xrightarrow{+32}$	'c'	→	99
'D'	→	68	$\xrightarrow{+32}$ $\xleftarrow[-32]{}$	'd'	→	100
⋮				⋮		
'X'	→	88		'x'	→	120
'Y'	→	89		'y'	→	121
'Z'	→	90		'z'	→	122

$$\begin{tikzcd} \text{upper} \arrow[r, shift left, "+32"] & \text{lower} \arrow[l, shift left, "-32"] \end{tikzcd}$$

print ((char) s[i]+32)

```
void  toggle (char s[]) {
        int  n = s.size()
        for (i=0 ; i<n ; i++) {
            if (s[i] >=65 && s[i] <=90) {
                print (s[i] +32)
            }
            else {
                print (s[i] -32)
            }
        }
}
```

char int

char ch = (char) 'a' + 3

↓

97 + 3

$\underline{\boxed{100}}$

(s[i] >=65 && s[i] <=90)

s[i] >= 'A' && s[i] <= 'Z'

Substring

↓

Contiguous subsequence of characters

1. Continuous part of string
2. Single char → substring
3. Entire string → substring

"abc" No. of substrings =6

"a" "b" "c" "ab" "bc" "abc"

Invalid → "ac" , "cb"

"bxcd"

No. of substrings $= \frac{n(n+1)}{2}$

4	b	→	"b"	"bx"	"bxc"	"bxcd"
3	x		"x"	"xc"	"xcd"	
2	c		"c"	"cd"		
1	d		"d"			
10						

len = 4 = n

no. $= \frac{\cancel{4}^{2}(5)}{\cancel{2}} = \underline{10}$

3. Check if given substring is palindrome.

$$\begin{array}{llllllllll} & & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ \text{char ch[8]} & : & n & e & w & \underline{m} & \underline{a} & \underline{d} & \underline{a} & \underline{m} \end{array}$$

int start = 3

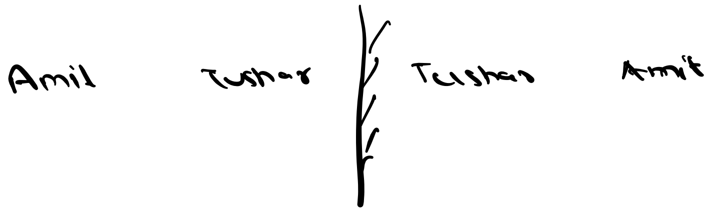
int end = 7

ans = True

Palindrome → sequence of characters that read same forward & backward

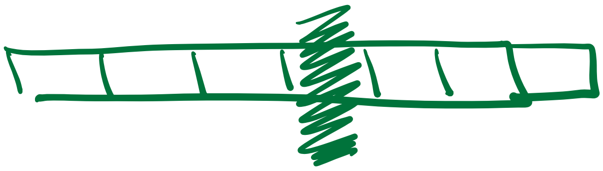
malayalam nayan aburva ✗

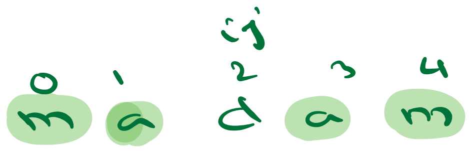
madam vikas

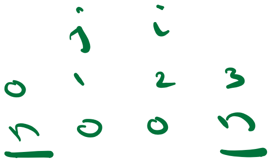


Palindromes → symmetrical around centre

Place a mirror at centre,
first half is symmetrical to second half

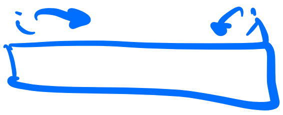






```
bool   isPalindrome (char s[], int st, int e) {
    int i = st , j=e ;
    while (i<j) {
        if ( s[i] != s[j])
            return false
        else {
            i++   j--
        }
    }
    return true
}
```

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



Given a string, calculate length of longest palindromic substring.

	0	1	2	3	4	5
s :	a	b	a	c	a	b

ans = 5

s : f e a c a b a c a b g f   ans = 7

s : a d a e b c d f d c b e t g g t e

ans = 9

Brute Force : Consider every substring
→ Check if palindrome
↓
Try to update ans

```
char str [ ]

int N = s.size()                    s ——→ e
int ans = 0

for (s = 0 ; s < N ; s++) {
    for (e = s ; e < N ; e++) {
        // s e of a substring
        if (isPalindrome(str, s, e)) {
            ans = max(ans, e-s+1)
        }
    }
}
```

TC: O(N^3)

SC: O(1)

1 substr → O(N)

N^2 substr → $N^2 \times N$

10:36

Optimisation: Think about a centre and expanding it

madam

malayalam

odd length

abba

raceecar

Even Length

Treat every char as centre

adaecbcdfdcbetggtc

abaabb aa

ans = ~~0~~ ~~1~~ ~~3~~ ~~4~~ 6

Odd Length

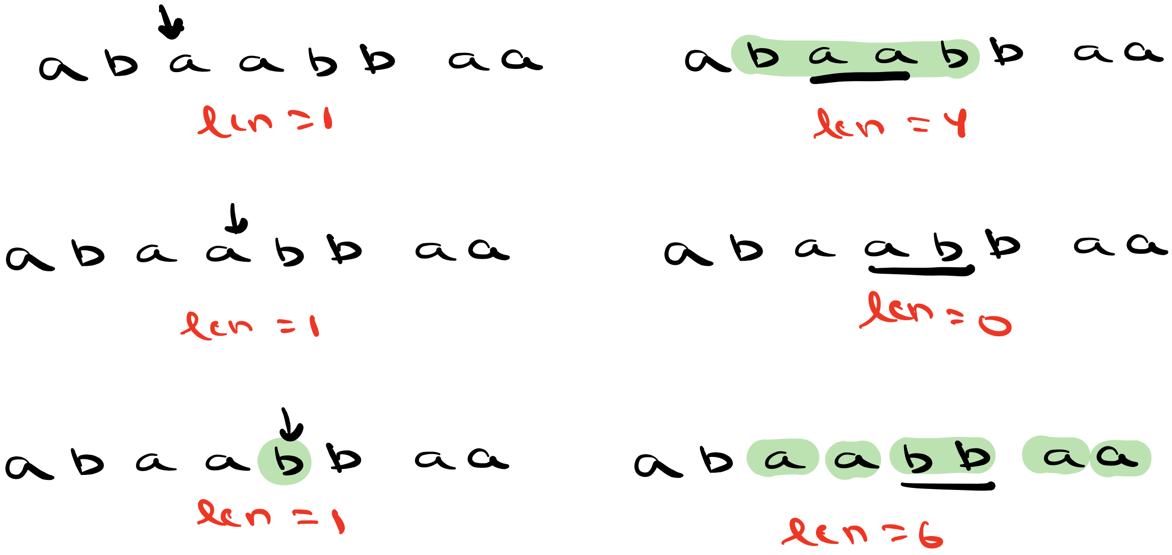
↓
abaabb aa
len = 1

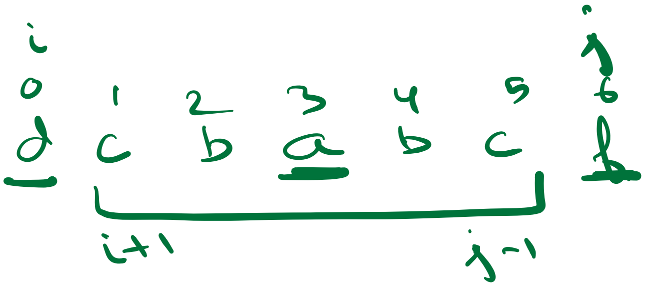
↓
abaabb aa
len = 3

Even Length

i j
abaabb aa
len = 0

abaabb aa
len = 0



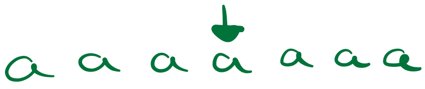


[a — b]

b-a+1

$$j-1-(i+1)+1$$

$$\boxed{j-1-i}-y+x$$



TC: $O(N^2)$

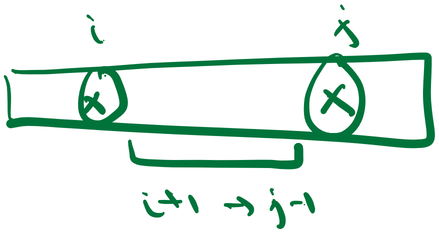
SC: $O(1)$

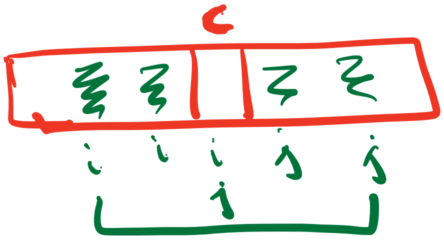
```
int longestPalindrome (char s[]) {
    int ans = 0
    int N = s.size()

    for ( c = 0 ; c < N ; c++) {

        # odd length
        int i = c , j = c
        while( i >= 0  && j < N) {
            if ( s[i] != s[j])
                break
            i--   j++
        }
        ans = max (ans, j-i-1 )

        # even length
        int i = c ; j = c +1
        while( i >= 0  && j < N) {
            if ( s[i] != s[j])
                break
            i--   j++
        }
        ans = max (ans, j-i-1 )
    }
}
```





$len = j-1 -(i+1)+1$

$= j-1 - i -\cancel{1} +\cancel{1}$

Immutability of string

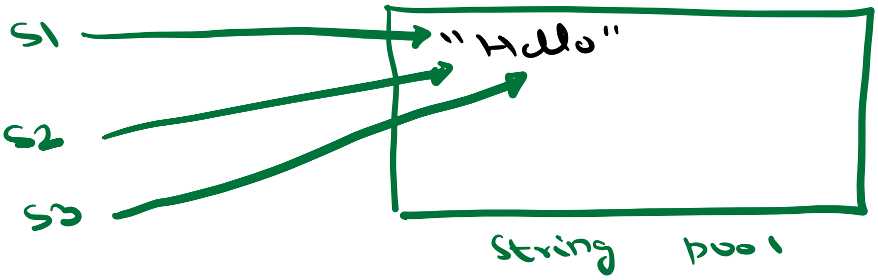
[a b]

b-a+1

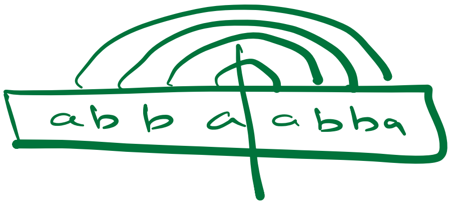
→ cannot change once built/assigned

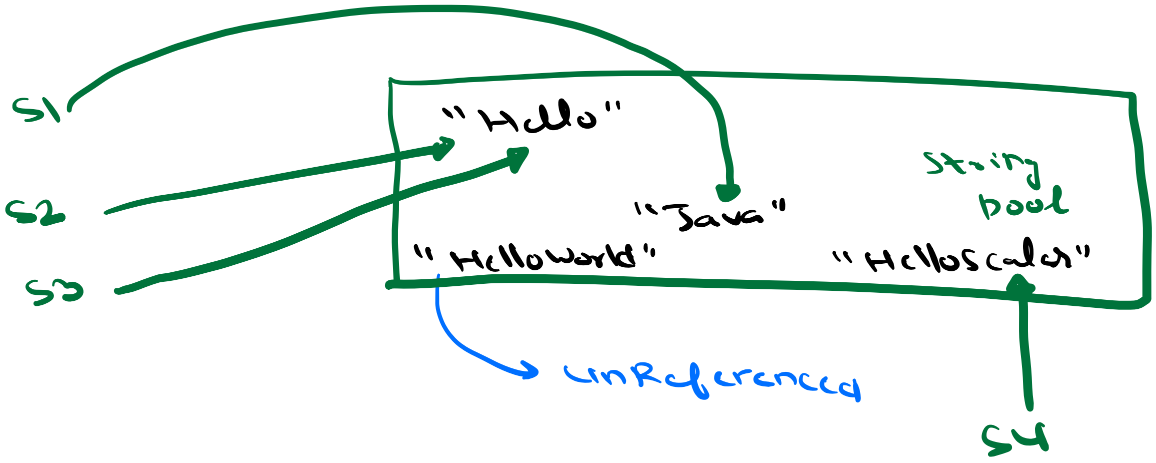
Lang → Java, C#, JS, Python, Go

strings are immutable, its value can't be changed



```
String s1="Hello"
string s2 ="Hello"
string s3 = s1
```





S1 = "Java'

S2. concat ("World")

String S4 = S2. concat ("Scala")

We need garbage collector to clean unreferenced strings.

Disadvantage

String S1 = " abcde"

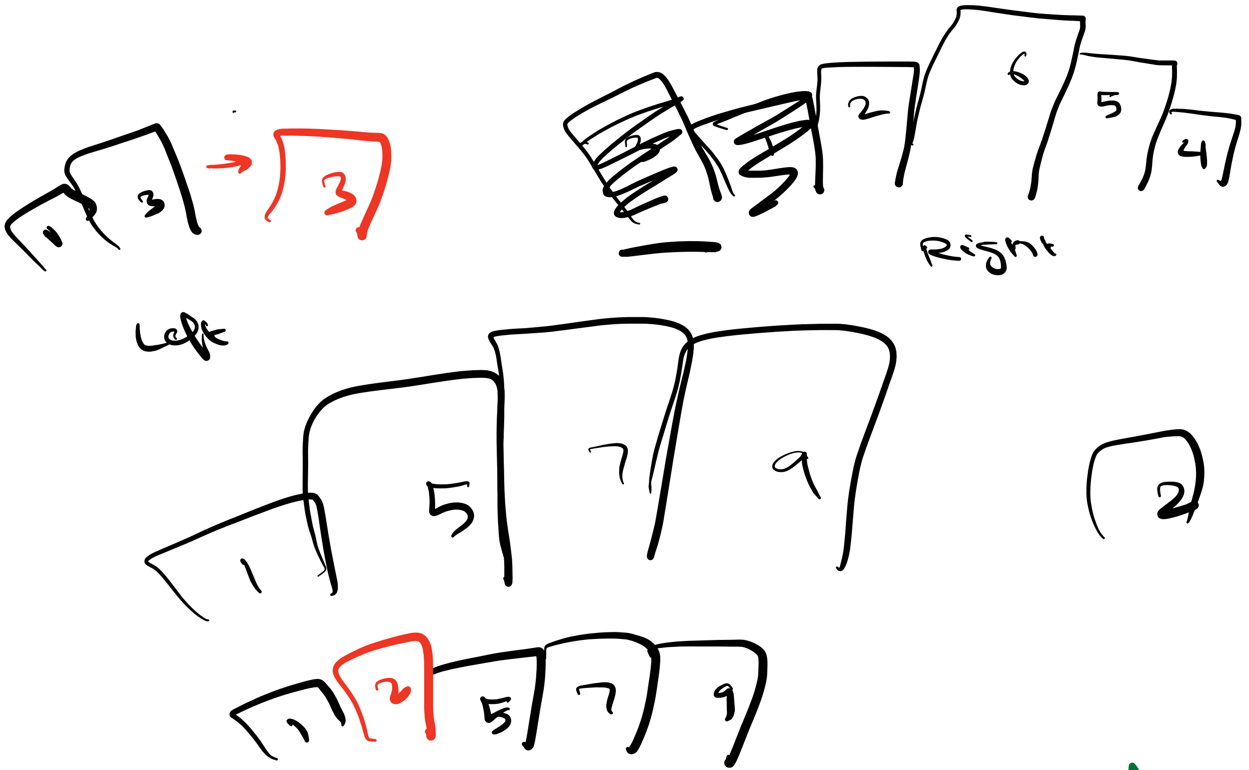
String S2 = S1. concat("z")   TC : O(N)

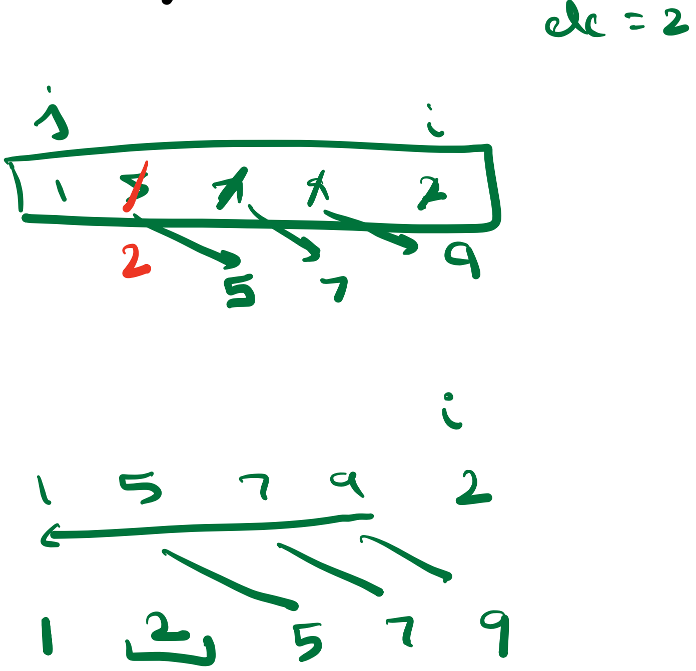
Mutable string → String builder
↓
array of chars

Hashing → hashmap / hashsets

Storing passwords

Designing databases , blockchain tech





① Selection

② Insertion

9, 7, 5, 4, 6, ②

1st

2 → 4 → 5 → 6