Strings

String :

- sequence of chars
- array of chars.
- collection of chars → {a, c, b}] → not same
  {a, b, c}

order is important

" ..... "
↑
string

Characters : ASCII values

" hello "
↑ ↑↑↑↑

'h', 'e', 'l', 'l', 'o'

~~'he'~~

'A'	- 65	$\xrightarrow{+32}$ / $\xleftarrow{-32}$	'a' - 97	'0'	- 48
'B'	- 66	$\xrightarrow{+32}$ / $\xleftarrow{-32}$	'b' - 98	'1'	- 49
'C'	- 67		'c' - 99	'2'	- 50
⋮			⋮		
'Z'	- 90		'z' - 122	'9'	- 57

'10' - It is not a single char

char ch = '9'

1 Byte

ASCII = 57

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

ch = '9' ASCII = 57

ch = ch + 8 → 57 + 8 = 65

print(ch)

'A'

```
char ch = (char) 65;
print(ch)
```

→ 'A'

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

– 'b'

```
int x = 'a';
print(x)
```

– 97

Question

Given a char array, toggle every char.

↳ lowercase ⇄ uppercase

Note: Inp. only contains alphabets.

ch[] = AnaConDa

output → aNAcONdA

aDgbHJe

AdeBhjE

```
void toggle ( char s[], n ) {

    for(i=0; i<n; ++i) {
        if( s[i] >= 65 && s[i] <= 90 ) {   // uppercase
            s[i] += 32
        }
        else {   // lowercase
            s[i] -= 32
        }
    }
}
```

single char

'A' → 65

'Z' → 90

32 → ('a' - 'A')

TC: O(N)

SC: O(1)

```
if (s[i] >= 'A' && s[i] <= 'Z') {
    s[i] += ('a' - 'A')
else
    s[i] -= ('a' - 'A')
```

Substring → contigous part of a string

1. Continous part of string
2. full string can be substring
3. Single char is also substring

string = "abc"

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

→ 6 substrings

"bxcd" n=4

total substrings → $\frac{n(n+1)}{2}$

$\frac{4 \times 5}{2} = 10$

Question

Check if a given substring is Palindrome or not?

left → right ≡ right → left

eg madam, naman, dad, abba

level, malayalam etc.

s = " a n a m a d a m s p e "

0 1 2 3 4 5 6 7 8 9 10

start, end

start = 3, end = 7

```
bool isPalin( char s[], start, end ) {
    i = start, j = end
    while ( i < j ) {
        if ( s[i] != s[j] ) {
            return false
        }
        i++, j--
    }
    return true
}
```

TC: O(N)

O(end - start)

Question

Given a string, calculate length of longest palindromic substring.

eg

a b a c a b

len = 5

a b c d e

len = 1

f e a c a b a c a b g f

len = 7

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

len = 9

Brute force

For all substrings, check Palindrome.

```
def longestPalin(char s[], n) {
    ans = 0
    for(i=0; i<n; ++i) {
        for(j=i; j<n; ++j) {
            if( isPalin(s,i,j) ) {
                len = j-i+1
                ans = max(ans, len)
            }
        }
    }
    return ans
}
```

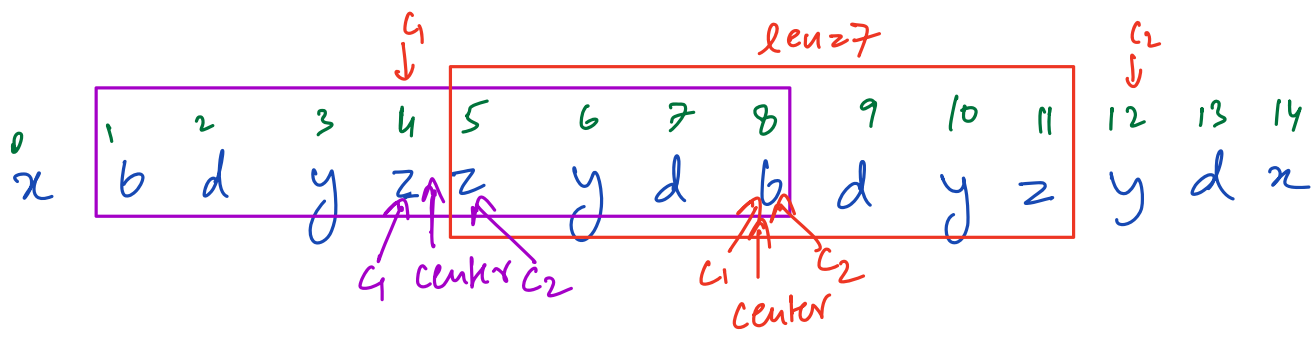
$\hookrightarrow O(N)$

$O(N^3)$

TC: $O(N^2)$

SC: $O(1)$

Idea



→ If the center of longest palindromic substring is given, then we can find length is $O(N)$

→ Since we don't know center,

take all possible centers $= N + N - 1$

$= O(N)$ centers

a b b a b

```
int expand( char s[], int c1, int c2)
    while ( c1>=0 && c2<n && s[c1] == s[c2] ){
        c1--, c2++
    }
    return c2-c1-1   <- len of longest palindromic
                        substring from center [c1, c2]
}
```

```
def longestPalin( char s[], n ){
    ans = 0

    for (i=0; i<n; ++i) { // odd length palindromes
        //center: i
        c1 = i, c2 = i
        ans = max(ans, expand(s, c1, c2))

    }

    for ( i=0; i<n-1; ++i ) { // even length palindromes
        //center: i, i+1
        c1 = i, c2 = i+1
        ans = max(ans, expand( s, c1, c2))

    }
    return ans
}
```

TC : $O(N^2)$

SC : $O(1)$