Array of characters ✓
Sequence of characters ✓
Group of characters ✗

(a b c)

abc ≠ bac

Characters (ASCII Values)

32

'a' → 97	'A' → 65	'0' → 48
'b' → 98	'B' → 66	'1' → 49
⋮	⋮	⋮
'z' → 122	'Z' → 90	'9' → 57

"007"

typecasting

```
char c = (char) 65
print (c)   // o/p → 'A'

char c = (char) ('a' + 1)
print (c) // o/p → 'b'

int x = 'a'
print (x)  // o/p → 97
```

97 + 1 = 98

Q→ Give a string (lowercase / uppercase alphabets). Print the string after flipping the case of every character.

Eg → "Hello"
o/p → "hELLO"

```
for i → 0 to (N-1) {
    if (s[i] >= 'a' && s[i] <= 'z') { // lower case
        print ((char)(s[i] - 32))
    } else {    // Uppercase
        print ((char)(s[i] + 32))
    }
}
```

TC = O(N)

SC = O(1)

Substring

a b c d x y t a p

d x y t → substring

b x c d

4
3
2
1
10

#substrings = $\frac{N*(N+1)}{2}$

String c = "x"

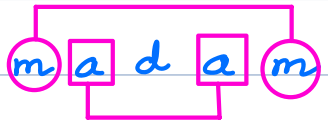
char a = 'x'

Q → Check whether the given string is a palindrome.

L → R = R → L

Eg → "racecar" o/p → true

"anar" o/p → false



```
l = 0    r = N-1
while (l < r) {
    if (s[l] == s[r]) {
        l++
        r--
    } else {
        return false
    }
} return true
```

"racecar"
0 1 2 3 4 5 6
r a c e c a r
l r

TC = O(N) SC = O(1)

Q → Find the length of longest palindromic substring.

Eg → "anamadam" Ans = 5

'a'

Eg → "feacabacabgf"

Ans = 7

Eg → "adaebcdfdcbetggte"

Ans = 9

Bruteforce →

```
ans = 0
for l → 0 to (N-1) {
    for r → l to (N-1) {   // l — r
        if (checkPalindrome(s, l, r))
            ans = max(ans, r-l+1)
    }
```

0 1 2
a b c

l=0 → a, ab, abc
1 → b, bc
2 → c

} return ans [l r]

[2 5] → 5 - (2-1) = 4

TC = O(N^3) SC = O(1)

"anamadam"

1) odd length

"0 1 2 3 4 5 6 7"
"anamadam" len = ~~4~~ ~~3~~ 5

single middle element → consider every char as mid & expand char by char.

2) even length → 2 mid points

"abcddccd" Ans = 4

```
ans = 0
for m → 1 to (N-1) {
    l = m-1   r = m+1
    len = 1
    while (l >= 0 && r <= N-1) {
        if (s[l] == s[r]) {
            len += 2
            l--   r++
        } else break
    } ans = max(ans, len)   // odd length
```

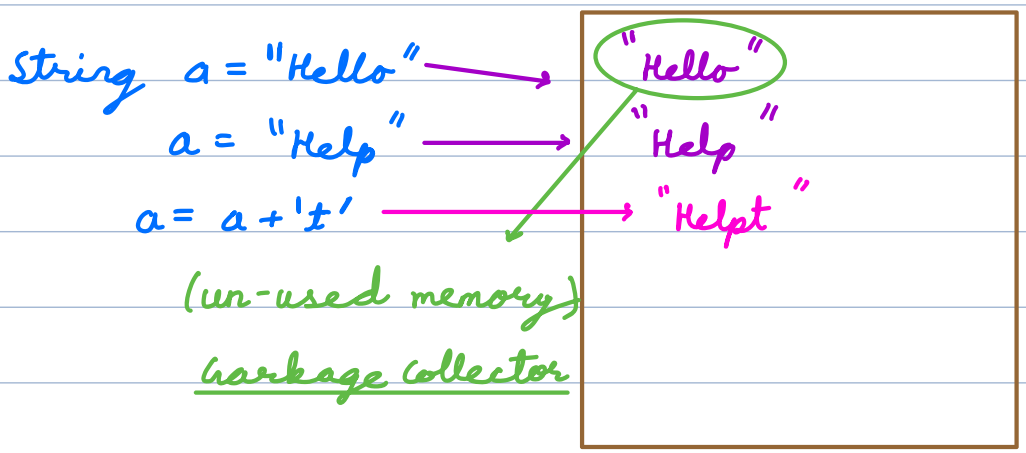
}

Similarly trove for ever len case.

TC = O(N²) SC = O(1)

String Immutability

Java / C# / JS / Python / Go → immutable | C / C++ → mutable



String Pool

int x = 10

x = 8 (same memory update)

String s = "abc"

s += 'd' ← TC = O(len of s)

s += 'e'

abc

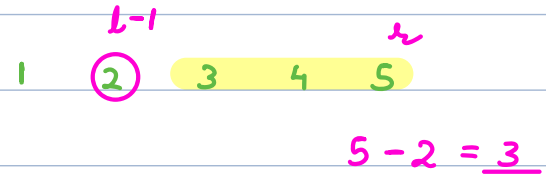
abcd

Create a string by appending N characters one by one → TC = O(N²)

→ string builder → TC = O(N)

mutable

```
String s = " "
for ∀c (char) in input {
    s = s + c
}
```



$5 - 2 = 3$