⇒ [1, 3, 5, 9, 14]   target = 12

$\frac{4 + 23}{2} = \frac{27}{2} = \boxed{13}$

STRINGS

like
a character array

$[\underbrace{'g'}_{x}, 'e', 'e', 'k', 's']$ size()

strings   "geeks"   length()

Strings are terminated with a special character ('\0')

'\n' → endline

'\t' → tab-characters

'\0' → string terminator

"geeks" → internally stored as "geeks\0"

"geeks"

"geeks" (\0) → $\boxed{\text{"ge"}}$

Int   'a'

We have a distinct ASCII value associated with each character.

char ch = 'a'

print ((int) ch) → Print the ASCII value of ch = 97.

check if a given character is lowercase English alphabet.

if (ch >= 97 && ch <= 122)

⇓

if (ch >= 'a' && ch <= 'z')

'a' → 97

⋮

'z' → 122

Check if Uppercase:

if (ch >= 'A' && ch <= 'Z')

check if digit

if (ch >= '0' && ch <= '9')

Convert a given lowercase character to corresponding uppercase.

if ch = 'b' ⇒ ch = 'B'

ch = 'y' ⇒ ch = 'Y'

Soln: ch = ch - 32

ch = ch - 'a' + 'A'

= ch - 97 + 65 = ch - 32

$\boxed{ch = \underline{ch - 'a'} + 'A'}$

ch	ch-'a'	ch-'a'+'A'
a	0	0 + 'A' = 'A'
b	1	1 + 'A' = 'B'
c	2	2 + 'A' = 'C'
d	3	3 + 'A' = 'D'
⋮	⋮	⋮
z	25	25 + 'A' = 'Z'

Convert uppercase to corresponding lowercase.

$\boxed{ch = ch - 'A' + 'a'}$

Palindrome: Something which is the same when read forward or backward.

"abba" → Palindrome

"abca" → Not Palindrome

"abab"

|1|2|3|2| → ×

|1|2|1| → ✓

Approach-1

1) Create a copy & reverse it.
2) Compare the original & reversed version.

TC = O(n)
Aux. space = O(n)

Approach-2

Two-pointer approach

"abba" → True

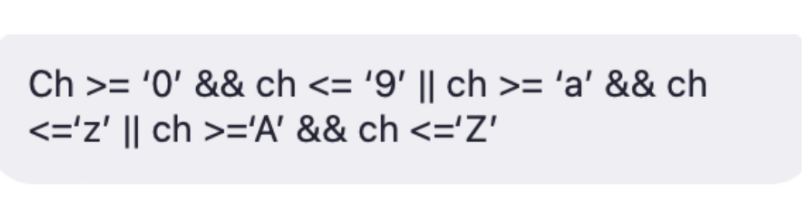
TC: O(n)
Aux. space: O(1)

"abca" → False

"A man, a plan, a canal: Panama" ⇒ True

"race a car"   'e' ≠ 'a' ⇒ False

"aba"



Ch >= '0' && ch <= '9' || ch >= 'a' && ch <='z' || ch >='A' && ch <='Z'

ANAGRAMS

s1 = "abcd"   s2: "abcdd"   ×

s1 = "abbc"   s2: "abcb"   ✓

TC

Sorting approach → O(n log n)

Hashmap - approach → O(n)

↳ storing freq. of all chars in the strings: O(n)

Comparing freq. of each character: O(26) = O(1)

[s1 = abbc
s2 = abcb]

s1

a → 1
b → 2
c → 1
d → 0
e → 6
⋮
z → 0

s2

a → 1
b → 2
c → 1
d → 0
e → 0
⋮
z → 0

	0	1	2	3		25
freq_s1:	5	7	3	0	---	7
freq_s2:	5	7	3	0	---	7

	0	1	2	3		25
char-freq	+5 -5	+7 -7	+3 -3	+0 -0	---	+7 -7

⇓

If all values in 'char-freq' are equal to zero in the end. Then both strings are anagrams.