CHAR ARRAYS

char ch[5];

→ 'a', 'b', 'c', ... 'z'
'A', 'B', 'C', ... 'Z'
'0', '1', '2', ... '9'

1 char → 1 byte

char name[100];
cin >> name; → parth

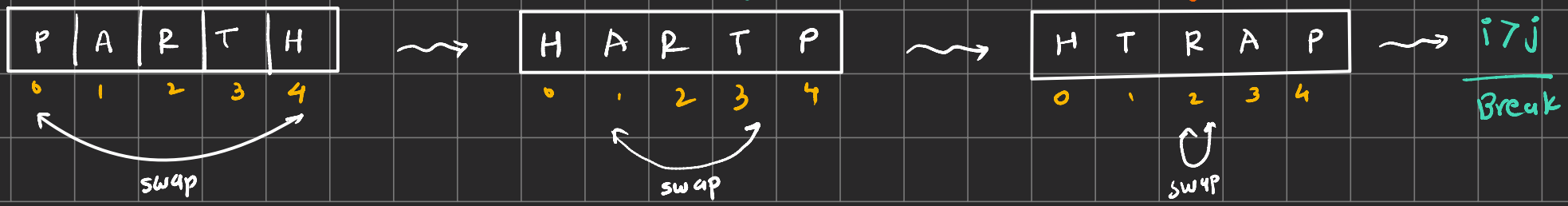
→ cout << int(name[5]);
→ 0 is ascii value of null

cin → space ' '
→ tab \t
→ newline \n
enter

Solve → cin.getline(ch, length)
20

→ strlen(ch)

→ Reverse string



→ palindrome → racecar

① input → racecar
reverse input → racecar
} compare → T / F

Tc = O(n+n) = O(2n) = On
Sc = O(n)
↳ bcz for reverse string !

② i, j

r	a	c	e	c	a	r
0	1	2	3	4	5	6

Tc = O(n)
Sc = O(1)

while (i<=j)
↳ if (arr[i] == arr[j])
i++
j--
Else
↳ not palindrome

→ uppercase lowercase —convert→ uppercase

lowercase char − 'a' + 'A' = uppercase char

→ 'c' − 'a' + 'A'
→ 99 − 97 + 65
→ 2 + 65
→ 67 → ascii of 'C'

→ lowercase lowercasechar = uppercasechar + 'a' − 'A'

→ if input is like Parth → PARTH, here we can put conditions like
if (ch[i] >= 'a' && ch[i] <= 'z')

✲ String

```
#include <string>
string str ;
```

⟶ `cin>> str ;`

`getline (cin, str)`

p	a	r	t	h	\0

↳ not accessable

→ `str.length()`

`.empty()` → is empty ? T/F

`.push_back('A')`

`.pop_back()`

`.substr(0,6)` (index, length)

parthPatel ⟶ parth

0 1 2 3 4 5 6 7 8 9

1 2 3 4 5 6

→ `str1.compare(str2) == 0` → exactly same

azzyy 'a ≠ b'

bzzyy a → b

97 < 98 → return (-1)

↳ for b → a

98 > 97

↳ return (+1)

i

l	o	v	e

→ in starting compare length len1 ≠ len2 so not same

l	o	v	e	r

→ compuring i == j

j

→ sentence.find(target) ⟶ return index from where target starts

string sentence = 'hello Jee kaise ho' ; string target = 'kaise' ;

0 1 2 3 4 5 6 7 8 9 10

target

```
string sentence = "hello Jee kaise ho saare";
string target = "Everyone";

cout << sentence.find(target);
if(sentence.find(target) == std::string::npos) {
        cout << "Not Found" << endl;
}
```

→ no position

→ replace (startingIndex , howmanycharsremove , replacingString)

```
string str = "This is my First Message";
string word = "Babbar";

str.replace(11, 5 , "Second");
cout << str << endl;
```

11 — 5 remove

→ This is my Second Message

→ str. erase (startingIndex , how many chars erase)

A	B	C	D	E	F	G	H	I	J
0	1	2	3	4	5	6	7	8	9

starting index → 3

1 2 3 4 5 → 5 char remove

→ A B C I J

☆ Remove All Adjacent Duplicates in string

(Adjacent: પાસ પાસ છે)

→ ans = " "

ans = "a" abbaca

ans = "ab"

here both are same so pop last char from ans

ans = "a" abbaca

ans = " " abbaca

ans = "c"

ans = "ca"

☆ Remove All occurances of a substring

main → | d | a | a | b | c | b | a | a | b | c | b | c |

part → | a | b | c |

daabcbaabcbc → dababc → dab

* Valid Palindrome II

a	b	c	a

remove one char and check that palindrome or not

a b c a i = j ⟶ no need to remove
i j

a b c a i ≠ j ⟶ remove one ⟶ remove i to make palind
i j or
remove j to make palind

a b c d ⟶ bcd
i i+1 j
check it is palindrome or not

a b c d ⟶ abc
i j-1 j
check it is palindrome or not

* Minimum time difference

["23:59", "00:00"] ⟶ [1439 + 0] mins

23 : 59
0 1 2 3 4

convert string to int } stoi("23")

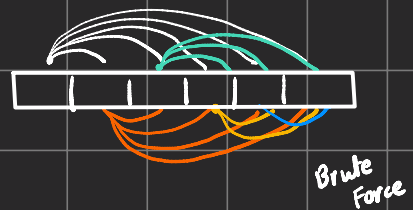
= 23 * 60 + 59
= 1439

12:10	10:15	13:15	17:20	18:00	19:47	23:59

converting to mins

730	615	795	1040	1080	1187	1439

sort Just to reduce TC



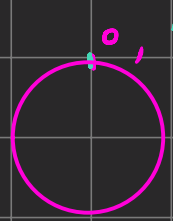
IMP test case

lastdiff = (min[0] + 1440) − min[n-1]

sorted

because in clock

circular difference



1440 min = 24 hr

615	730	795	1040	1080	1187	1439

$diff_1$ $diff_2$ d_3 d_4 d_5 d_6 → find min from diff all

end → minimum of (min, lastdiff)

★ Palindromic Substrings (MIMP)

a	b	c

→ substrings find → contigious sich ई

a	b	c

i j

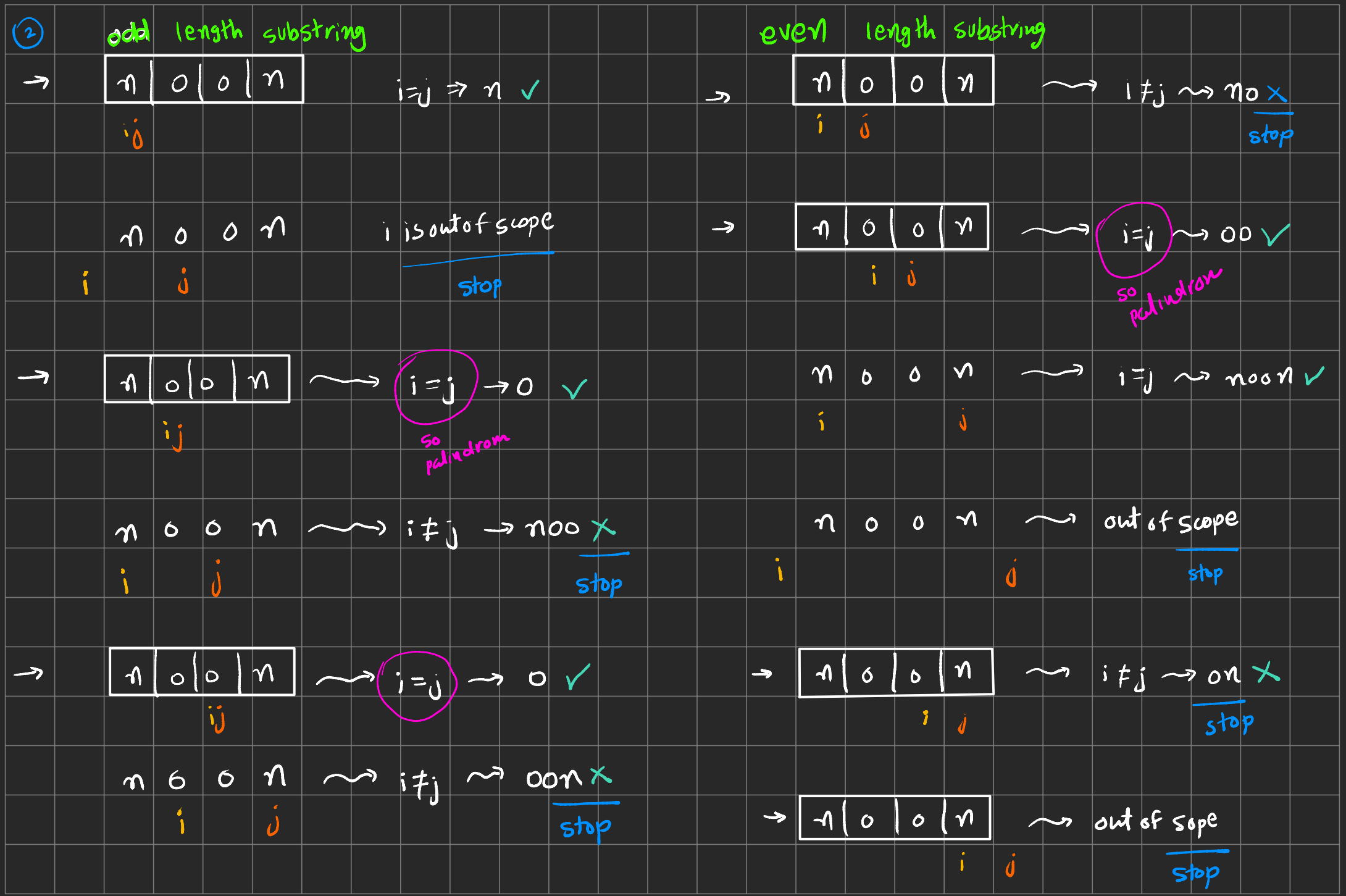
① a | b | c

ab | bc

abc

find palindrom from all substrings → a, b, c

TC: $O(n^3)$



→ | n | o | o | n | → i == j → n ✓

i j

TC = O(n²)

n o o n → j is out of scope

i j

stop

★ How to pass custom comparator into sort function

string s = "babbar"

sort (s.begin() , s.end()) → according to lexicography, ascii values → aabbr → ascending order

what if I need into decreasing order → rbbaa

```
bool cmp(char first, char second) {
        return first > second;
}

int main() {
  string s = "babbar";
  sort(s.begin(), s.end(), cmp);
```

descending

→ first character ki first aayi gi if condition is becoming true then → rbbaa

can be used with INT value like bool (int first, int sec) return first > sec

→ return first < second → increasing order

→

"love"	"babbar"	"rahul"	"sandeep"
= l+o+v+e = 12+15+22+5 = 54	= b+a+b+b+a+r = 2+1+2+2+1+18 = 26		

⟶ we can make custom comparator according to the total sum of all characters

→ using auto

```
vector<string> ans{"love", "babbar", "rahul", "sandeep"};

for( auto x: ans ){
    cout << x << " ";
}
cout << endl;
```

NOTE: while founding maximum number, int maxi = INT_MIN
— — minimum number, int mini = INT_MAX

✱ Valid Anagram

s = "anagram" t = "nagaram" ⟶ True

s = "cat" t = "rat" ⟶ False

anagram of a string is another string that contains the same characters, only the order of characters can be different. For example, “act” and “tac” are anagrams of each other.

① sorting - sorting according to ascii value

s → aaagmnr } same
t → aaegmnr }

Tc. O(n log n)

```
cook@pop-os:~$ ascii -d
   0 NUL    16 DLE    32      48 0    64 @    80 P    96 `   112 p
   1 SOH    17 DC1    33 !    49 1    65 A    81 Q    97 a   113 q
   2 STX    18 DC2    34 "    50 2    66 B    82 R    98 b   114 r
   3 ETX    19 DC3    35 #    51 3    67 C    83 S    99 c   115 s
   4 EOT    20 DC4    36 $    52 4    68 D    84 T   100 d   116 t
   5 ENQ    21 NAK    37 %    53 5    69 E    85 U   101 e   117 u
   6 ACK    22 SYN    38 &    54 6    70 F    86 V   102 f   118 v
   7 BEL    23 ETB    39 '    55 7    71 G    87 W   103 g   119 w
   8 BS     24 CAN    40 (    56 8    72 H    88 X   104 h   120 x
   9 HT     25 EM     41 )    57 9    73 I    89 Y   105 i   121 y
  10 LF     26 SUB    42 *    58 :    74 J    90 Z   106 j   122 z
  11 VT     27 ESC    43 +    59 ;    75 K    91 [   107 k   123 {
  12 FF     28 FS     44 ,    60 <    76 L    92 \   108 l   124 |
  13 CR     29 GS     45 -    61 =    77 M    93 ]   109 m   125 }
  14 SO     30 RS     46 .    62 >    78 N    94 ^   110 n   126 ~
  15 SI     31 US     47 /    63 ?    79 O    95 _   111 o   127 DEL
```

② Counting

↓ frequency table

s = "anagram"

a → 3
g → 1
m → 1
n → 1
r → 1

t = "nagaram"

a → 3
g → 1
m → 1
n → 1
r → 1

if both freq. table are same then

→ we can go with hashmaps

→ Here

→ total 256 Ascii values

→ for first string's chars

do, arr[a] ≈ arr[(int)a] ++

→ second strings char

do (--)

} in end if not 0 then not....

★ Reverse only letters

s = "ab-cd" ⟶ "dc-ba"

① two Pointer Approach with some rules

T	e	s	t	1	n	g	-	L	e	e	t	=	c	o	d	e	-	Q	!

l h

① if (s(l) → alphabet & s[h] → alphabet) ⇝ swap (s[l], s[h]), l++, h++

elseif (s[l] → not alphabet) ⇝ l++

else ⇝ h--

* isAlphabet → a-z → [97 - 122] and A-Z → [65 - 90] ascii ch

→ isalpha (ch) → T/F c++ func STL

T	e	s	t	1	n	g	-	L	e	e	t	=	c	o	d	e	-	Q	!

k l h h k

Q e d o S e - T !

* longest common prefix

["codehelp" , "codenothelp"]

f	l	o	w	e	r
i	i	i			

f	l	o	w
j	j	j	

f	l	i	g	h	t
k	k	k			

not same

if i = j = k → go above

* Reverse vowels of string

a,e,i,o,u

s = "IceCreAm" → "AceCreIm"

I	c	e	C	r	e	A	m
l		l			h		
A		e			e	I	

l++, h--

If (l==h → vowel) swap

If l → not vowel → l++

h → not vowel → h--

☆ Isomorphic strings

s = "egg" t = "add" ⟶ True

e ⟷ a
g ⟷ d
} unique mapping

s = "foo" t = "bar" ⟶ False

f ⟷ b
o ⟷ a
o ⟷ r ✗ unique mapping not possible

Here o is already mapped with a
and we are trying to map o to r again not possible

→ Sol. hash {256}

255 → total 256

hash [s[i]] = t[i]

s = | p | a | p | e | r |
i

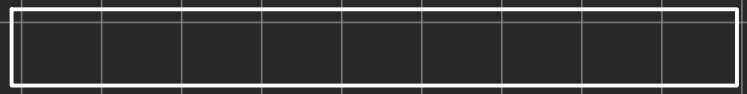
t = | t | i | t | l | e |
i

hash [p] = t
hash [112] = 116

p → t
a → i
p → t
e → l
r → e

p → t
p → t
} mapping with same char means okey

But if there is something like p → x then False bcz p is already mapped with t that's why



255 → total 256

isMapped (t[i]) = true

egg / add	foo / bar	paper / title	badc / kikp
e → a	agg	f → b	boo	p → t	taper	b → k	kadc
g → d	add	o → a	baa	a → i	tiper	a → i	kidc
		o → r → you cannot do bcz o → a		p → t	titer	d → k	
	True	False		e → l	titlr	↳ you cannot do it bcz k is mapped with b already	
				r → e	title	False	

★ Reorganize String

so that same chars are not side by side adjacent

s = "aab" ⇝ "aba"

s = "aaab" ⇝ " "

abaa ×

aaba ×

s = aaabc ⇝ abaca

s = aaabb ⇝ ababa

① using priority queue (n.log n)

② greedy

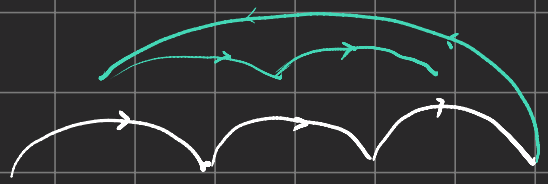
① Most occurencent character and fit it non-adjacently in one go (one iteration)

② fill the rest characters

→

a a a b b e f

a → 3
b → 2
e → 1 f → 1



a b a e a f b

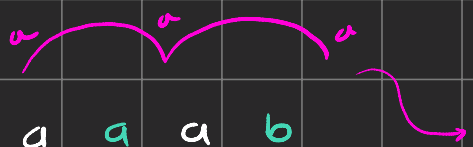
0 1 2 3 4 5 6

→ a a a b b b e f g g

a → 3
b → 3
e → 1
f → 1
g → 2

a b a e a f b g b g

0 1 2 3 4 5 6 7 8 9



→ aaab

a → 3
b → 1

a a a b

0 1 2 3

When I am placing most occurent character it should be placed in one go like

a a a

0 1 2 3

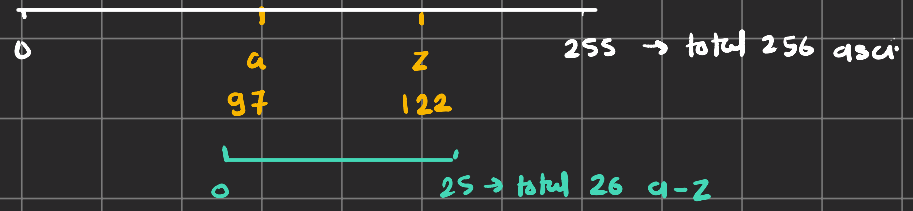
but here

it is not possible so FALSE

① count hash

② Try to place most occurent in one go ⇝ If (not possible) → return false

③ Place other chars with one index gap



[s[i] - 'a'] → 0

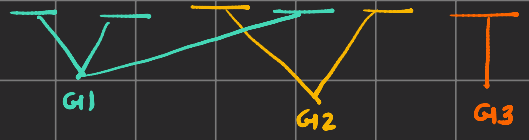
'a' - 'a' → 0

✱ Group Anagrams

strs = ["eat", "tea", "tan", "ate", "nat", "bat"] ⇝ [["bat"], ["nat", "tan"], ["ate", "eat", "tea"]]

① eat, tea, tan, ate, nat, bat

sorted → aet, aet, ant, aet, ant, abt



Map → (first: key, second: value)

	key	value
①	aet	eat, tea, ate
②	ant	tan, nat
③	abt	bat

```cpp
//#include <map>
map<string, vector<string>> mp;

for (auto str : strs) {
    string s = str;
    sort(s.begin(), s.end());
    mp[s].push_back(str);
}

vector<vector<string>> ans;
for (auto it = mp.begin(); it != mp.end(); it++) {
    cout << "key: " << it->first << " | value: ";
    for (const auto& str : it->second) {
        cout << str << " ";
    }
    cout << endl;

    ans.push_back(it->second);
}
```

key → `it->first`, value → `it->second`

```
key: abt| value: bat
key: aet| value: eat tea ate
key: ant| value: tan nat
```

TC ⇒ str whole string → N

sorting n log n → K log K

map → ignore now

O(N K log K)

SC ⇒ O(N K) → bcz storing ans

- N → length of strs. so N = 6
- K → length of longest ele K = 3
- ans → • length = 6 • longest ele's length under ans = 3

② without sorting

hash [256] = {0}

string A, string B

hashmapA == hashmapB } anagram

key	value
hash [256]	vector<string>
① e-1, a-1, t-1	eat, tea, ate
② t→1, n→1, a→1	tan

① "eat" creating hashmap above ② for "tea" same hash key is coming so store value of tea

```cpp
std::array<int, 256> hashFunc(string s) {
    std::array<int, 256> hash = {0};
    for (int i = 0; i < s.size(); i++) {
        hash[s[i]]++;
    }
    return hash;
}

vector<vector<string>> groupAnagrams(vector<string>& strs) {
    map<std::array<int, 256>, vector<string>> mp;
    for (auto str : strs) {
        mp[hashFunc(str)].push_back(str);
    }

    vector<vector<string>> ans;
    for (auto it = mp.begin(); it != mp.end(); it++) {
        ans.push_back(it->second);
    }

    return ans;
}
```

new way to create Array

std :: array < dataType, size > arrName ;

= {0}

TC = O(N.K)

SC = O(N.K) + Map size

★ Longest Palindromic substring

S = "babad" ⟶ "bab" or "aba"

① Sub strings of string

② extract Palindromic ones

↳ find longest from them

```cpp
bool isPalindrome(string& sub, int start, int end) {
    while (start < end) {
        if (sub[start] != sub[end]) {
            return false;
        }
        start++;
        end--;
    }
    return true;
}

string longestPalindrome(string s) {
    string ans = "";
    for (int i = 0; i < s.size(); i++) {
        for (int j = i; j < s.size(); j++) {
            if (isPalindrome(s, i, j)) {
                string t = s.substr(i, j - i + 1);
                ans = t.size() > ans.size() ? t : ans;
            }
        }
    }
    return ans;
}
```

① substrings

• b a b a b → a
ab

• b a b a b → b

Find index of first occurance

s = "sadbutsad"

t = "sad"

s a d b u t s a d
0 1 2

0 th index

→ L e t s a d u
0 1 2 3 4 5 6

s a d

s a d

s a d

s a d

sliding window

→ if initial character of t matched then match other chars of t

★ String to Int (atoi) / run length encoding

" __ lakshay 4139 "

```cpp
#include <iostream>
#include <cstdlib> // Required for atoi
#include <string>

int main() {
    std::string str1 = "12345"; // C++ std::string
    const char* str2 = "   -6789"; // C-style string (const char*)

    // Using stoi with std::string (str1)
    int num1_stoi = std::stoi(str1); // Converts "12345" to 12345
    std::cout << "Using stoi, str1: " << num1_stoi << std::endl; // Output: 12345

    // Using atoi with C-style string (str2)
    int num2_atoi = atoi(str2); // Converts "   -6789" to -6789
    std::cout << "Using atoi, str2: " << num2_atoi << std::endl; // Output: -6789

    // Using atoi with str1 converted to C-style string (str1.c_str())
    int num3_atoi = atoi(str1.c_str()); // Converts "12345" to 12345
    std::cout << "Using atoi, str1.c_str(): " << num3_atoi << std::endl; // Output: 12345
```

① ignore leading white space

② determine sign

③ if digit found after finding sign - integer prepare

④ till next nondigit char found

⑤ special handling of if num is out of range so return INT_MIN or INT_MAX

→

a	a	b	b	c	d	e	e
0	1	2	3	4	5	6	7
prev	i						

index

s → ~~a~~ ~~a~~ bb c dee

a 2

cout = 1

prev == i → cout++

prev ≠ i → s[index] = prev

cout > 1 then s[index + 1] = cout

cout = 1

a a b b c d e e

prev i

s → a a bb c d e e

index

a 2

INT to char

digit + '0'

a a b b c d e e

prev i

cout = 2

a a b b c d e e

p i

s → a 2 bb c d e e

index

b 2

a a b b c d e e

p i

s → a 2 b 2 c d e e

index

c

cout = 1 so do not store it on (index + 1)

a a b b c d e e

p i

s → a 2 b 2 c d e e

index

d

a a b b c d e e

p i

a a b b c d e e

p i

s → a 2 b 2 c d e e

index

e 2

Int to Roman

27 → X X V I I

10 + 10 + 5 + 1 + 1 = 27

58 → num >= RomanValue

58 >= 50 ⟶ 50 + 8

L + 8

num = 58 − 50

= 8

8 >= RomanValue

8 >= 5 ⟶ 5 + 3

V + 3

num = 8 − 5

= 3

3 >= RomanValue

3 >= 1 ⟶ While (num >= 1) ⟶ add symbol I

num = num − value

largest to smaller

Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

3333 = 1000 + 1000 + 1000 + 333 → num = 3333 - 3000
M M M + 333 = 333

333 = 100 + 100 + 100 + 33 → num = 333 - 300
C C C + 33 = 33

33 = 10 + 10 + 10 + 3 → num = 33 - 30
X X X + 3 = 3

3 = 1 + 1 + 1 → num = 3 - 3
I I I = 0

1994 = 1000 + 994 → num = 1994 - 1000
M = 994

994 = 100 + ... 9 times + 94

BUT ans is MCMXCIV

900 → CM = |100 - 1000| = 900

like IV = |1 - 5| = 4

} so we need new list with specials

① Map Iterate (larger to small)

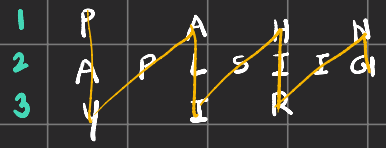
```
while ( num >= value[i] )
{
    roman = roman + symbol
    num   =  num - value[i]
}
```

```
string romanSymbols[] = {"M", "CM", "D", "CD", "C", "XC", "L", "XL", "X", "IX", "V", "IV", "I"};
int values[] = {1000,900,500,400,100,90,50,40,10,9,5,4,1};

string ans = "";

for(int i=0; i<13; i++){

    while(num>=values[i]){
        ans = ans + romanSymbols[i];
        num = num - values[i];
    }
}
```

M = 1000
CM = 900
D = 500
CD = 400
C = 100
XC = 90
L = 50
XL = 40
X = 10
IX = 9
V = 5
IV = 4
I = 1

✱ Zig Zag conversion

S = "PAYPALISHIRING" rows =



→ PAHNAPLSIIGYIR

take 3 vector string and store values like this

0 → PAHN
1 → APLSIIG
2 → YIR

✡ Largest Number

sort
decrease

54321
largest num

9534303

9534330 → Ans

lexicography sort

```
bool mycom ( string a, string b)
return  a>b
```

this say that
30 > 3
but we need
something that
say 3 > 30

solve

a = 30 , b = 3

string t1 = 303 = a+b
t2 = 330 = b+a

return t1 > t2 ← True

means t1 કો પહેલો રખો

303 > 330 F

t2 કો પહેલો રખો