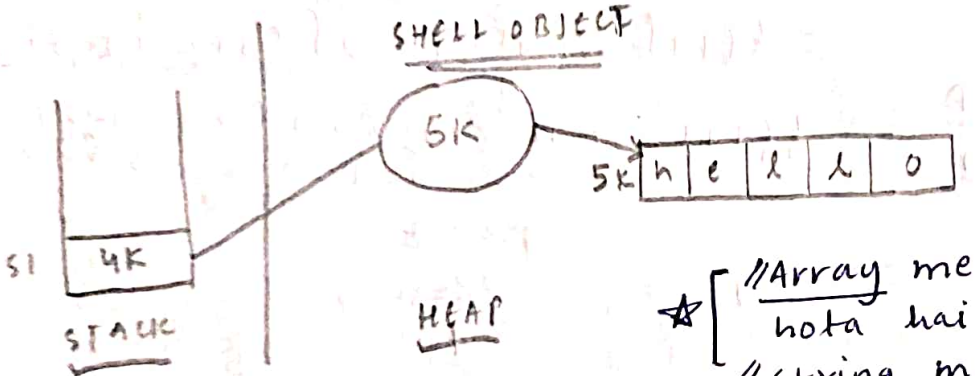
② INTRO TO STRING, STRING-BUILDER, ARRAY LIST

```
p s v m(s[] a){
{String s1 = "hello";   ] Define & Declare
 Syso(s1);
```

```
{Scanner s = new Scanner(System.in);
 String s1 = s.next();  → Space tak hi read krega!
 String s2 = s.nextLine(); → Enter tak read krega!
```

Input :
hello World ⟶ hello
hello huys ⟶ hello huys




```
{String s = s.nextLine();   → abcdef
 Syso(s);  → abcdef
 Syso(s.length()); → 6
```

```
{char ch = s.charAt(2);
 Syso(ch); ⟶ c
```

```
{for(int i=0; i<s.length(); i++){
 char ch = s[i]; //WRONG SYNTAX
 char ch = s.charAt(i); //RIGHT
   Syso(ch);
 }
```

a b c d e f

```
// s[0] = 'b';
// s1.charAt(0) = 'b';   ] NOT POSSIBLE!
```

★ [//Array me Length ek datamember hota hai!
//string me length ek function hota hai! function ki trah call krna hoga!
//Array me agar ek element/char chaiye toh hum s[0] krty thy!
//string me s.charAt(0) krengy

//Array me het krkey set bhi krskty hai!

//String me sirf aur sirf het krty hai set nahi

```
s += "world";
Syso (s);
```
String Addition

∴ Syso (10 + 20 + "hello" + 10 + 20);

30 + "hello" + 10 + 20

"30hello" + 10 + 20

"30hello10" + 20

30hello1020 Output

```
String s = "hello"; //h0, e1, l2, l3, o4
Syso (s.substring (1,3)); el
          (0,4)); hell
          (0,5)); hello
          (0)); hello
          (3)); ello
          (0,0)); Blank milega
          (2,1)); Error Ayega
```

First index kabhi bhi last index se bada nahi hoga!

s.substring (FIRST INDEX (i) , LAST INDEX (j));

↳ Yeh (i) se (j-1) tak ke element dega

Let us say we have to print all substrings of abcd.

s = "a b c d"
0 1 2 3

01 → a
02 → ab
03 → abc
04 → abcd

12 → b
13 → bc
14 → bcd

2-3 → c
2-4 → cd

3-4 → d

// jab i = 1 tab
j = 2
∴ (i) and (j = i+1)

i [0 to l-1]
j [i+1 to l]

```
String s = "abcd";
for (int i=0; i<s.length(); i++){
  for (int j=i+1; j<=s.length(); j++){
    Syso (s.substring (i,j));
  }
}
```

```
String s = "abc def ghi jkl";
String[] parts = s.split (" ");
for (int i=0; i<parts.length; i++){
    Syso (parts[i]);
}
```

abc
def
ghi
jkl

// Yeh kisse bhi split krelega (,) se bhi

INTRO TO ARRAYLIST → DEKHNE ME ARRAY JAISA NAHI HOGA, PERFORMANCE ARRAY JAISI DEGA!

int [] arr = new int[5] → ek baar 5 size ka bana diya toh 5 size ka hi rahta hai!

ArrayList<Integer> list = new ArrayList<>(); → ISSE Arraylist Declare and Define hota hai!

↳ shurruwat me toh 0 size hai lekin size badana chayengy toh usme chejy add hojyengi! ☆ → need basis pe chalega

```
p s v m (s [] a){
[ArrayList<Integer> list = new ArrayList<>();
 System.out.println(list + "->" + list.size());] → [] → 0   (empty hai size 0 hai)

[list.add(10);
 list.add(20);
 list.add(30);] Adding in Arraylist
 Syso(list + "->" + list.size()); → [10,20,30] → 3

[list.add(1,1000); // Bichme insert krne ke liye   (1st index pe 1000 daal do!)
 Syso(list + "->" + list.size()); → [10,1000,20,30] → 4

[int val = list.get(1); → use of get function to get a particular value
 Syso(val);

[list.set(1,2000); (1st index ko 2000 set krdo!)
 Syso(list + "->" + list.size()); → [10,2000,20,30] → 4

[list.remove(1); → remove this element!
 Syso(list + "->" + list.size()); → [10,20,30] → 3

// set value ko update ya change krta hai!
// add value (new) ko insert krta hai!

[ArrayList<String> list2 = new ArrayList<>();
 l2.add("Hello");
 ____("Bello");
 ____("Cello");
 Syso(list2 + "->" + list2.size()); → [Hello, Bello, Cello] → 3
```

```
for (int i = 0; i < list.size(); i++){
    int val = list.get(i);
    Syso(val);
}
```

size ke basis pe loop → 10, 20, 30

```
for (int val: list){
    Syso(val);
}
```

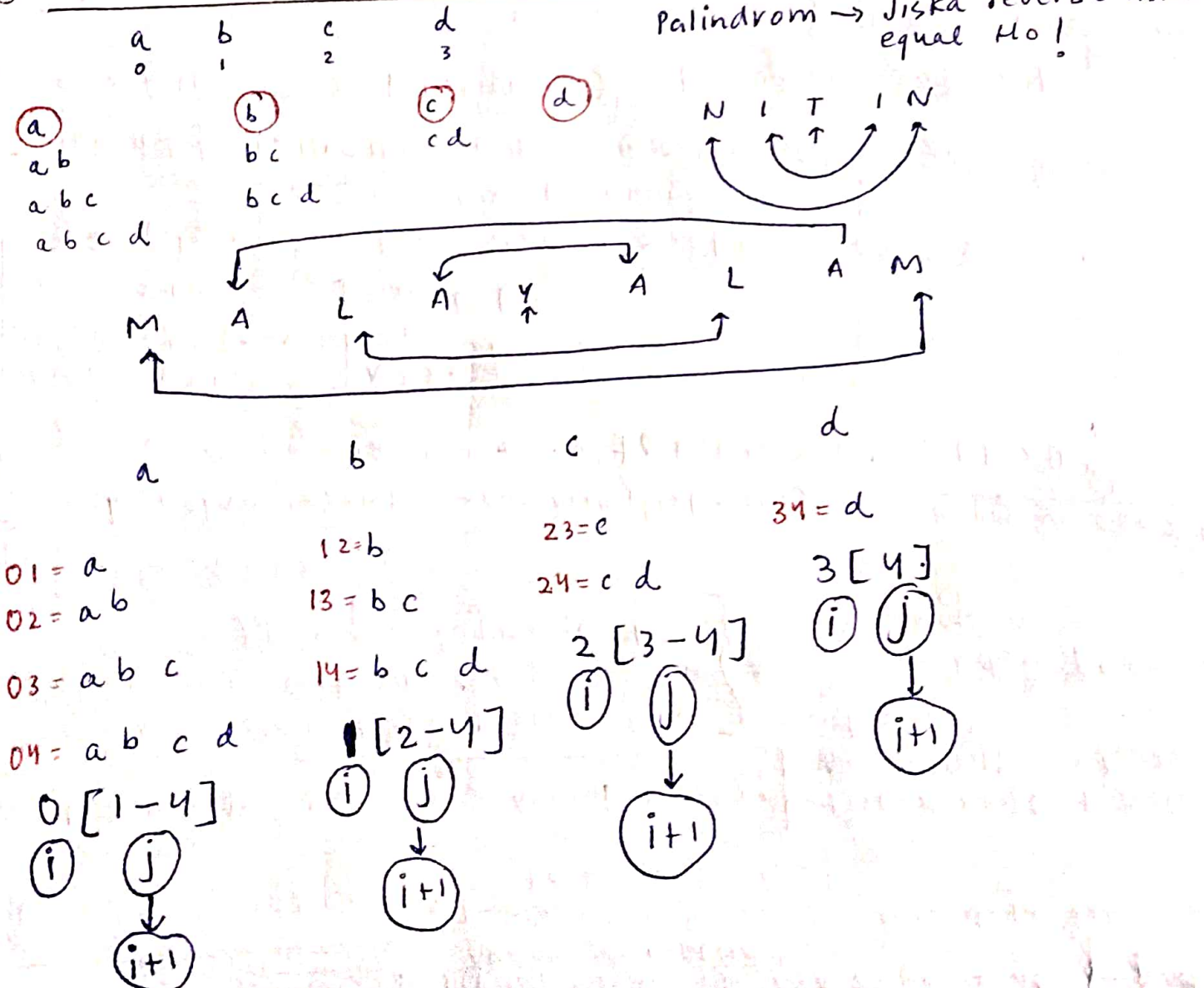
loop → 10, 20, 30

Agar intention sirf content print krna hai toh loop ki jarurat nahi hai!

```
Syso(list); → [10,20,30]
```

Print All Palindromic Substrings

Palindrom → Jiska reverse uske equal Ho!



```
p s v m(s [] a) {
Scanner s = new Scanner(System.in);
for(int i = 0; i < s.length(); i++){
  for(int j = i+1; j <= s.length(); j++) {
    string sub = s.substring(i, j);
    boolean isPalindrome = ISPalindrome(sub);
    if (isPalindrome == = true) {
       Syso(sub);
    }
  }
```

```
p s  boolean IsPalindrome (string sub) {
boolean flag = true;
int left = 0;          → i = starting point
int right = sub.length()-1;   → j = end point
while ( left < right) {
char chleft = sub.charAt(left);
char chright = sub.charAt(right);
if (chleft != chright) {
    flag = false;
    break;
 }
left ++;
right --;
}
return flag;

}
```

Pseudo Code

isPalindrome(String):
① Initialize 2 pointers (i) from 0 and (j) from string.length-1
② Now loop runs untill i<=j :
  (a) if character at index (i) is not equal to character at index (j), then return false.
  (b) Increment (i) by 1 and decrement (j) by 1
③ Return true.

Solution(String):
① Run a outer loop (i) from 0 to str.length-1
  (a) Run inner loop nested loop (j) from i+1 to str.length
  (b) Check if for Palindrom → yes → Print it

Time Complexity : $O(n^3)$

we are running :
outer loop : 0 to n-1 → O(n)
Inner loop : i to n-1 → O(n)
Generating substring : i to j → O(n)

from i to j, checking whether it is a palindrome

SPACE COMPLEXITY : O(n)

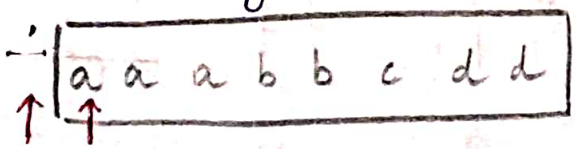
STRING COMPRESSION (Leetcode [Medium])

a a a b b c c c d d a e e f

function 1: a b c dae f

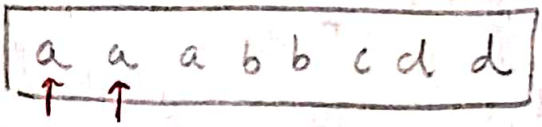
function 2: a3 b2 c3 d2 a e2 f

Let us say we have this string | a a a b b c d d |



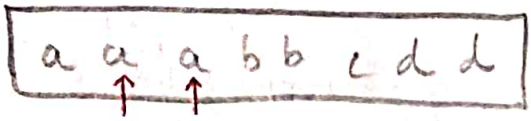
i - 1 < 0 ⇒ ADD

res_str = "" + 'a' = "a"



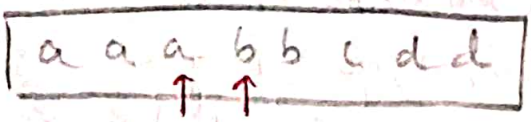
str[i-1] = str[i] ⇒ DUPLICATE

res_str = "a"



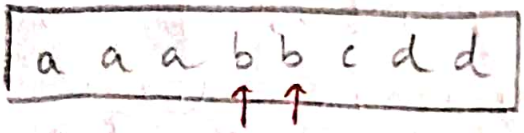
str[i-1] = str[i] ⇒ DUPLICATE

res_str = "a"



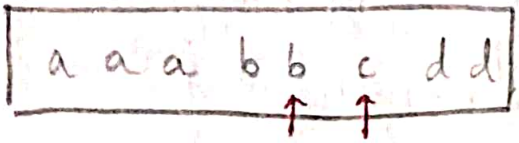
str[i-1] ≠ str[i] ⇒ ADD

res_str = "a" + "b" = "ab"



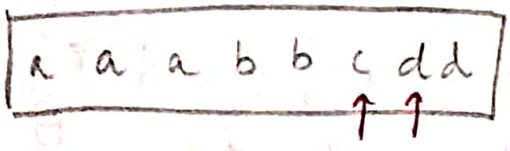
str[i-1] = str[i] ⇒ DUPLICATE

res_str = "ab"



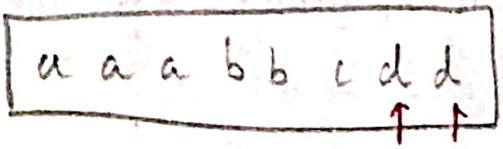
str[i-1] ≠ str[i] ⇒ ADD

res_str = "ab" + "c" = "abc"



str[i-1] ≠ str[i] ⇒ ADD

res_str = "abc" + 'd' = "abcd"



str[i-1] = str[i] ⇒ Duplicate

res_str = "abcd"

↓

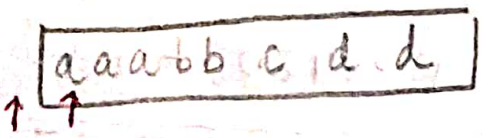
ANSWER

COMPRESSION !

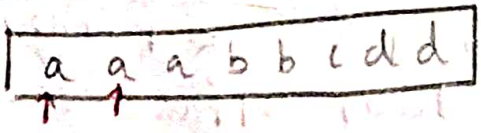
```
p s v m(s[]a){
 Scanner s = new Scanner(system.in);
 String str = s.next();
 Syso (compression1(str));
 ____ (compression2(str));
 }

p s string compression1 (String str){
 String s = str.charAt(0)+ " ";

for(int i=1; i < str.length(); i++){
   char curr = str.charAt(i);
   char prev = str.charAt(i-1);

   if (curr != prev) {
      s += curr;
    }
  }
  return s;
}
```

Input string : "a a a b b c d d" → Output = ?
Counter variable count = 0
ans = " "

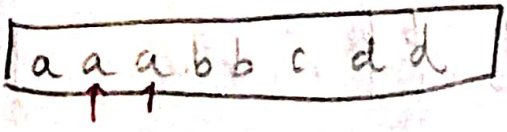
COMPRESSION 2



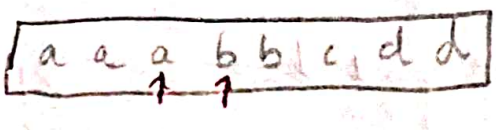
i-1 < 0 ⇒ first char.
⇒ append char & make count = 1
ans = " " + 'a' = "a" , count = 1



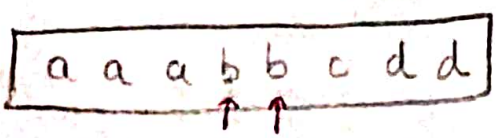
s[i-1] = s[i] ⇒ Increment counter
ans = "a" , count = 2



s[i-1] = s[i] ⇒ Counter ↑
ans = "a" , count = 3



s[i-1] ≠ s[i] ⇒
append prev count, curr char, make count = 1
ans = "a" + 3 + 'b' = "a3b" count = 1



s[i-1] = s[i] ⇒ Counter ↑
ans = "a3b" count = 2

aaabbcdd

s[i-1] ≠ s[i]
⇒ append prevcount, currchar, make count = 1
ans = "a3b" + 2 + 'c' = "a3b2c" count = 1

aaabbcdd

s[i-1] ≠ s[i], but count = 1
⇒ only append current char
ans = "a3b2c" + 'd' = "a3b2cd" count = 1

aaabbcdd

s[i-1] = s[i], Counter ↑
ans = "a3b2cd" counter = 2

Now after loop terminates,
count > 1, hence append count to output string.

ans = "a3b2cd" + 2 = "a3b2cd2" Result

```
public String compression2(String str) {
    String s = str.charAt(0) + "";
    int count = 1;
    for (int i = 1; i < str.length(); i++) {
        char curr = str.charAt(i);
        char prev = str.charAt(i-1);
        if (curr == prev) {
            count++;
        } else {
            if (count > 1) {
                s += count;
                count = 1;
            }
            s += curr;
        }
    }
    if (count > 1) {
        s += count;
        count = 1;
    }
}
```

TC: O(n)
SC: O(n) including spaces
O(1) without spaces

ans = a3b2cd3
count = ✗✗✗✗✗✗3

⇒ Start me a lekey chalenge
⇒ count ko 1 rakenge
⇒ index 1 se chalenge, agar same hai char toh count badega
⇒ Agar same nahi hai toh phele count add hoga string phir count 1 hojegga fir char add hoga string me.... so on
⇒ last me loop end ke baad agar count 1 se greater hai toh vo add hoga string me.

② STRING INTERNING AND IMMUTABILITY

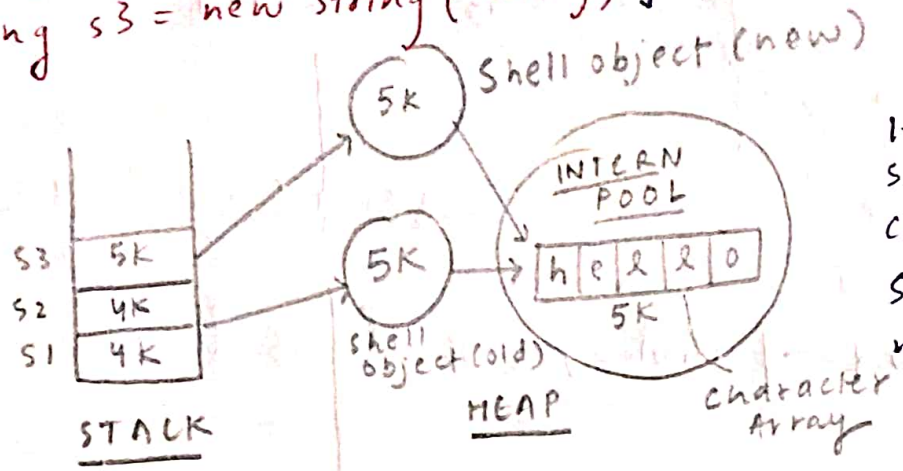
```
String s1 = "hello";
String s2 = "hello";
String s3 = new String("hello");
```

Teeno ka content same hai

Agar phele se hello bana hua hai toh hum usse point krengy

Interning

It is a method of storing only one copy of each distinct String values, which must be immutable.



Why Interning → To optimize space.

↓

IMPLICATIONS

① Equals → isme dam hota hai character by character check krne ka!

== → comparison me kabhi == nahi krna. use equals.

∴ s1 == s2 → yeh stack me (4K) dekh raha hai! → Address check krta hai bs!

↓ TRUE

s1 == s3 → BECOZ (==) address check krta hai!

↓ False ∴ {4K ≠ 5K ?

★ (==) → yeh smajhdar nahi hai! → just check on stack and not the content.

```
Syso (s1 == s2); ✓ True
Syso (s1 == s3); X False
Syso (s1.equals(s2)); ✓ True
```

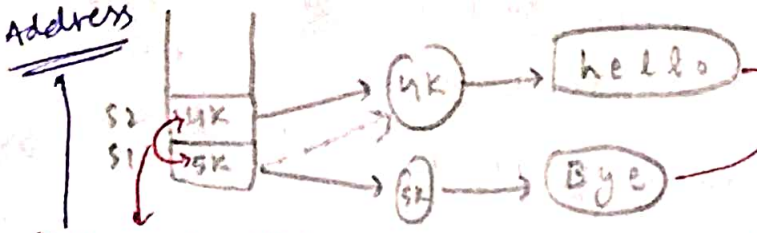
yeh bahut smart hota hai → First checks the stack (if equal return true, otherwise checks the content in heap).

2nd IMPLICATION

IMMUTABILITY → Interning ki wajah se string immutable hoti hai!

→ Hum string me koi bhi change nahi kr sakty!

★ Reference is mutable but instance is not.



Reference can be change becoz it's mutable

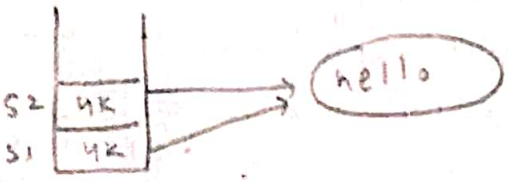
Inme koi bhi parivartan nahi ayega!

```
String s1 = "hello";
        s2 = "hello";
        s3 = "bye";
```

Instance cannot be changed becoz it's immutable.

∴ Address can be changed but content cannot be.

WHY IMMUTABILITY? → It is becoz of Interning



Agar humare pass `s1.setChar(1,'d')` hota toh hello converted dello

Par (S1) ke sath-sath (S2) bhi badal jata!

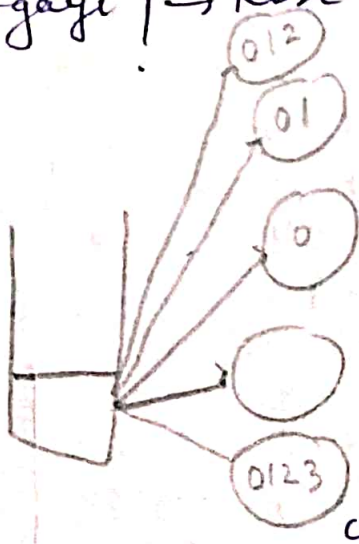
↳ Issey developer ko dikkat hogi!

IMPACT?

Performance of Strings slow hogayi! → Kese?

```
String s = "";
for(int i=0; i<=n; i++){
    s += i;
}
```

∴ Yeh O(n²) hai O(n) nahi! → KU? →



1+2+3+4+...+n

$= \frac{n(n+1)}{2}$

$= O(n^2)$

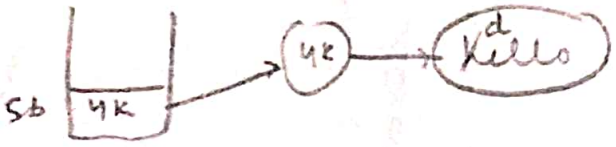
Har baar naya Shell object banega usme purane ka content copy hoga fir new add hoga

∴ address change hoga!

⑧ STRING BUILDER → USAGE AND Performance → string jaise hai but isme func hota hai change krne ke liye!

```
p s v m (S [] a){
StringBuilder sb = new StringBuilder("hello");
Syso(sb); → hello

char ch = sb.charAt(0); // get
Syso(ch); → h

sb.setCharAt(0,'d'); → Yeh strings me nahi hota! // update
Syso(sb); → dello
```

String's ke instance immutable hoty hai!
But StringBuilder me change hojata hai!




```
sb.insert(2,'y'); // insert
Syso(sb); → deyllo

sb.deleteCharAt(2); // remove
Syso(sb); → dello
```

```
sb.append('g'); // last me add krne ke liye append hota hai!
Syso(sb); → dellog
```

```
int n = 100000; → long start = System.currentTimeMillis();
String s = "";
for(int i=0; i<n; i++){
    s += i;
}
long end = System.currentTimeMillis();
long duration = end - start;
Syso(duration);
```

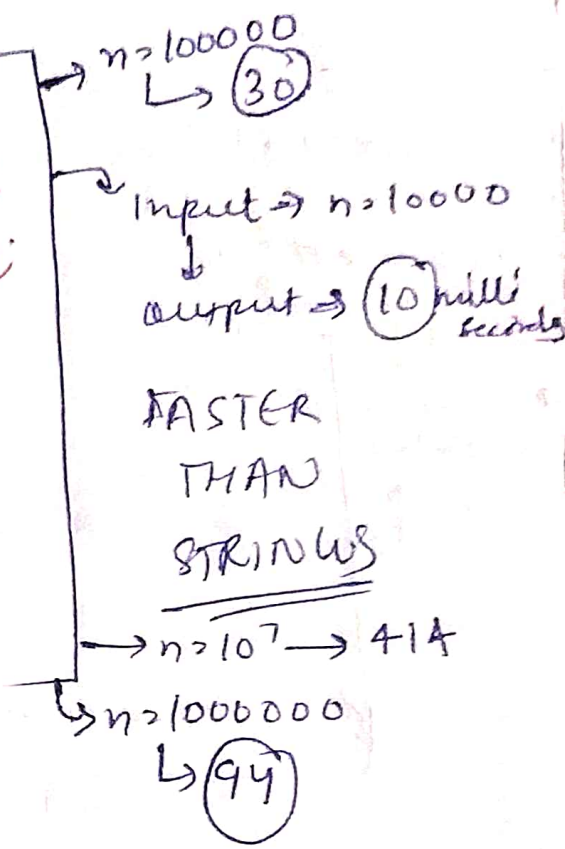
input n=100000 → TLE

input → n=10000 → 181 milliseconds

STRING WALA SCENE

When we use StringBuilder

```
p s v m(S[]a) {
int n = 10000;
long start = System.currentTimeMillis();
StringBuilder sb = new StringBuilder();
for(int i=0; i<n ; i++) {
        sb.append(i);
}
long end = ____________________;
long duration = end - start;
Syso(duration);
}
```



TEST QUESTION

Q

n = 5

1	2	3	4	5
11	12	13	14	15
21	22	23	24	25
16	17	18	19	20
6	7	8	9	10

+2n, +2n, -n, -2n

- Phele structure banegy !
- Phir values fill krengy !

→ i <= n/2 (upper half)
→ i == n/2+1 (middle row)
→ Analysis
→ i > n/2+1 (lower half)

Columns me +1 ka increment hai !
↳ Agar mai thar row ka starting point nikal lu ! fir +1 increment krna hai bus

val of first row first index is
val = 1

n = 6

1	2	3	4	5	6
13	14	15	16	17	18
25	26	27	28	29	30
31	32	33	34	35	36
19	20	21	22	23	24
7	8	9	10	11	12

+2n, +2n, +n, -2n, -2n

i < n/2 (upper half)
i == n/2 (middle row)
i > n/2 (lower half)

n → odd (5) → uh → i <= n/2, mr → i == n/2+1, lh → i > n/2+1 → Odd case
n → even (6) → uh → i < n/2, mr → i == n/2, lh → i > n/2 → Even case

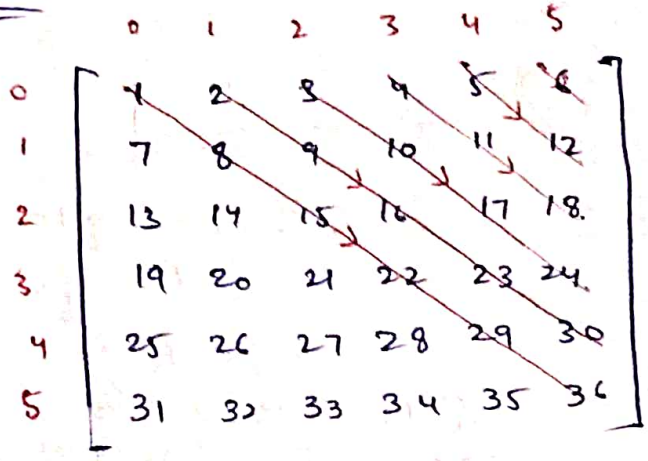
```
if (n%2 != 0) {
  if (i<=n/2) {
    val += 2*n;
  } else if (i == n/2+1) {
    val -= n;
  } else {
    val -= 2n;
  }
} else {
  if (i<n/2) {
    val += 2*n;
  } else if (i == n/2) {
    val += n;
  } else {
    val -= 2*n;
  }
}
```

Code

```
psvm (s[] a) {
  Scanner s = new Scanner(System.in);
  int n = s.nextInt();
  int val = 1;
  for (int i=1; i<=n; i++) {
    int ival = val;
    for (int j=1; j<=n; j++) {
      Syso (ival + "\t");
      ival++;
    }
    Syso ();
    //update value of val for next row
  }
}
```

8 DIAGONAL AND WAVE TRAVERSAL

n = 6



maps →

	0	1	2	3	4	5
0	0,0	0,1	0,2	0,3	0,4	0,5
1	1,1	1,2	1,3	1,4	1,5	
2	2,2	2,3	2,4	2,5		
3	3,3	3,4	3,5			
4	4,4	4,5				
5	5,5					

For every (gap) (diagonal) we need to manage starting point only

```
for (diag → 0 ⟶ n-1) {
                diag++
    for (i → 0, j → diag, j < n, i++, j++) {
        sysn(arr[i][j]);
    }
}
```

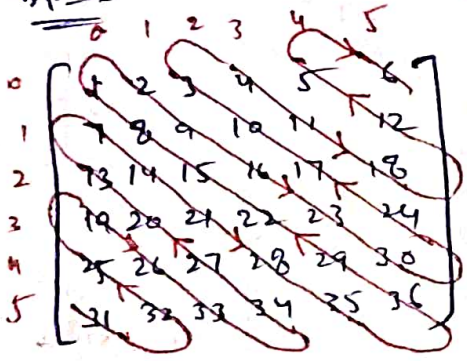
```
i → 0, j → diag
while (j < n) {
    i++, j++;
}
```

i → increasing
d → decreasing

Ab humne issi ques ko extent krdiya

	i	d	i	d	i	d	i	d	i	d	i
diag →	0	1	2	3	4	5	6	7	8	9	10
0	0,0	0,1	0,2	0,3	0,4	0,5	0,1	3,5	0,3	1,5	0,5
1	1,0	1,1	1,2	1,3	1,4	1,5	1,2	2,4	1,4	0,4	
2	2,0	2,1	2,2	2,3	2,4	2,5	2,3	1,3	2,5		
3	3,0	3,1	3,2	3,3	3,4	3,5	3,4	0,2			
4	4,0	4,1	4,2	4,3	4,4	4,5	4,5				
5	5,0	5,1	5,2	5,3	5,4	5,5					
	even: i = n-1-diag, j = 0	odd: i = n-1, j = diag					even: i = 0, j = diag-(n-1)	odd: i = 2n-2-diag, j = n-1			

n = 6



∴ For the even values of diagonals → increasing

Jab bhi diagonal even hoga → ek ek se badega

Aur

Jab bhi diagonal odd hoga → ek ek se ghatega → decreasing

Starting point dundana hai

i	j
5,0	
3,0	
1,0	

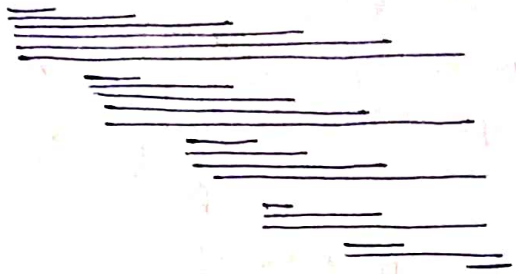
i = n-1-diag
j = 0 → column fix

i	j
5,1	
5,3	
5,5	

column varies
Row fix

KADANES ALGORITHMS → used in maximum subarray sum

6	-3	4	5	-8	1
0	1	2	3	4	5



subarray → continous subpart of array

↳ any subset of (i, j) is subarray where the subarray contains these element

i, i+1, i+2, ... j

↳ all continous

[∴ i <= j] → this also is condition

i ∈ (0, n-1)

j ∈ (0, n-1)

```
i j
0 0  (6)
0 1  -3+6 (3)
0 2  4-3+6 (7)
0 3
0 4
0 5
```

```
i j
1 1
1 2
1 3
1 4
1 5
```

```
i j
2 2
2 3
2 4 (1)
2 5
```

```
i j
3 3
3 4
3 5
```

```
i j
4 4
4 5
```

```
i j
5 5
```

$O(n^3) \longrightarrow O(n^2)$

Brute force $O(n^3)$

```
∴ max = -∞
for(i=0; i<n; i++) {  → Op1
  for(j=i; j<n; j++) {  → Op2
    sum = 0;
    for(int k=i; k<=j; k++) {  → Op3
      sum = sum + arr[k];
    }
    if(sum > max) {
      max = sum;
    }
  }
}
```

$O(n^2)$

Prefix - sum

ps → sum

ps[i] arr[0] → arr[i]

ps	6	3	7	12	4	5

∴ ps[5] → arr[0] ... arr[5] ✗

ps[4] + arr[5] ✓

$ps[i] = ps[i-1] + arr[i]$

0 - i - 1

[2 - 5] → ps[5] - ps[1]

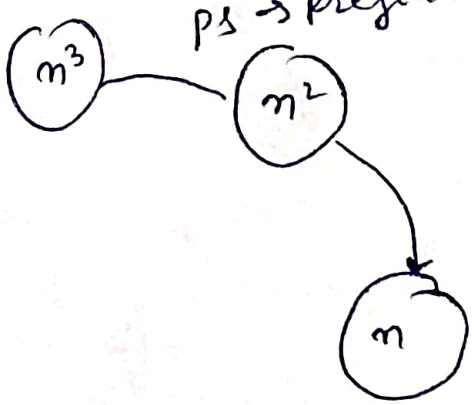
2 + 3 + 4 + 5

$(i,j) - ps[j] - ps[i-1]$

↳ if i = 0, then return ps[j]

```
if (i != 0)
  sum = ps[j] - ps[i-1]
else
  sum = ps[j]
```

ps → prefix sum



TOGGLE CASE

pepCODing ⟶ PEPcodING

'p' − 'a' = 'P' − 'A' ⟹ 'p' = 'P' − 'A' + 'a'

'P' = 'A' + 'p' − 'a'

Jitn durr small p small a se hoga utna hi capital P Capital A se dur hoga!

FORMULA

∴ lower case character = 'a' + upper case character − 'A'

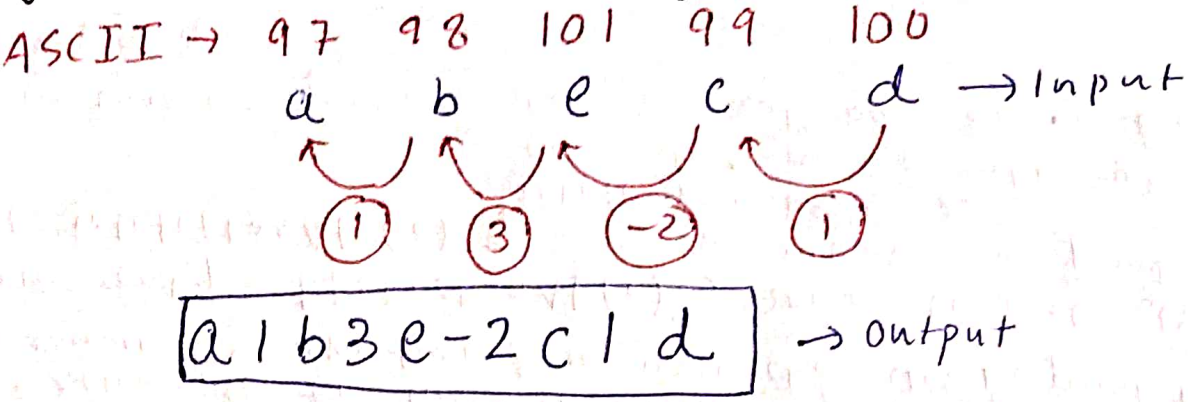
upper case character = 'A' + lower case character − 'a'

```
p s string toggleCase (s str){
StringBuilder sb = new StringBuilder(str);
for(int i=0; i<sb.length(); i++) {
  char ch = sb.charAt(i);
 if (ch>='a' && ch<='z'){
char uch =(char) ('A'+ ch - 'a');
 sb.setCharAt(i, uch);
} else if (ch >='A' && ch <='Z') {
  char lch= (char)('a'+ ch - 'A');
    sb.setCharAt(i, lch);
  }
}
  return sb.toString();
}
p s v m (S [] a) {
  Scanner s = new Scanner (System.in);
   String str = s.next();
   System.out.println(toggleCase(str));
}
```

→ Upper Case Conversion

→ Lower Case Conversion

String with Difference of Every Two Consecutive Characters



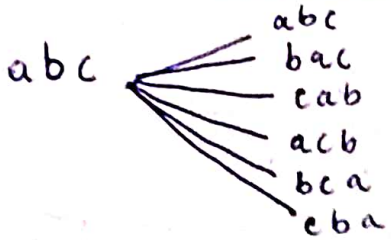

```
p s String solution(String str){
 StringBuilder sb = new StringBuilder();
  sb.append(str.charAt(0));
 for(int i=1; i< str.length(); i++){
    char curr = str.charAt(i);
    char prev = str.charAt(i-1);
    int gap = curr-prev;
    sb.append(gap);
    sb.append(curr);
  }
  return sb.toString();
}
p s v m(s[] a){
 Scanner s = new Scanner(System.in);
 String str = s.next();
  Syso(solution(str));
}
```

Time Complexity: O(n)

Space Complexity: O(n)

Print all Permutations of a String Iteratively

permutations



∴ abc → length = 3

∴ L, no. of permutations = 3! = 6

3	0	a b c ⇒ a
2	0-0	a b c ⇒ b
1	0-0	a b c ⇒ c
		∴ abc

3	1	a b c ⇒ b
2	0-1	a b c ⇒ a
1	0-0	a b c ⇒ c
	0-0	∴ bac

3	2	a0 b1 c2 ⇒ c
2	0-2	a0 b1 c ⇒ a
1	0-0	a b c ⇒ b
	0-0	∴ cab

3	3	a0 b1 c2 ⇒ a
2	1-0	a b0 c ⇒ c
1	0-1	a b c ⇒ b
	0-0	∴ acb

3	4	a0 b1 c2 ⇒ b
2	3-1	a0 b c ⇒ c
1	0-1	a0 b c ⇒ a
	0-0	∴ bca

3	5	a0 b1 c2 ⇒ c
2	1-2	a0 b1 c ⇒ b
1	0-1	a0 b c ⇒ a
	0-0	∴ cba

```
p s v solution(String str) {
    int n = str.length();
    int f = factorial(n);

    for(int i = 0; i < f; i++)   // Yeh loop 0 se (n-1) tak hi chalega kuki upar dekh i ke liye 0 se 5 chala hai!
    {
        StringBuilder sb = new StringBuilder(str);   // stringBuilder me string ko catch kiya kuki hume delete aur update krna hai.
        int temp = i;   // (i) ko disturb nahi krengy kuki (i) loop ko control krega!
        for(int div = n; div >= 1; div--) {
            int q = temp / div;   // Quotient
            int r = temp % div;   // Remainder
            // Aab remainder ke hisab se decide hoga ki kya print hona hai jo remainder aya hai hum ussi character ko print krty thy!
            System.out.print(sb.charAt(r));   // remainder wala character nikala aur print krdia!
            sb.deleteCharAt(r);   // delete bhi krdia vo original string me rahna nahi chaiye!
            // Jo iss baar ka quotient hai vo agli baar ka no. ban jata hai!
            temp = q;
        }
        sysO();
    }
}
```

```
p s int factorial (int n) {
    int val = 1;
    for (int i=2; i<=n; i++){
        val *= i;
    }
    return val;
}

p s v m (s [] a) {
 Scanner s = new Scanner(System.in);
 String str = s.next();
 Solution (str);
  }
```

Time complexity: $O(n! + n)$ ⇒ $O(n!)$

outer loop → 0 to n!
inner loop → 0 to n

Space complexity: $O(n)$

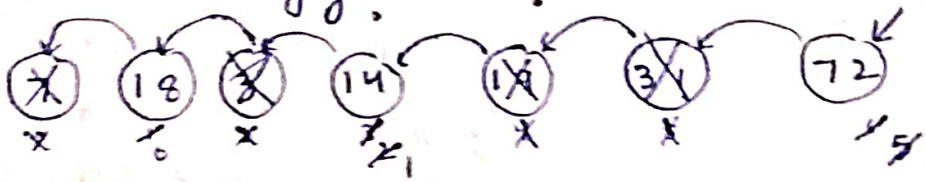
REMOVE ALL PRIME

0	1	2	3	4	5	6	7	8	9	10
8	2	13	16	24	17	32	19	4	3	7

After Removing Primes

0	1	2	3	4	5	6	7	8	9	10
8	2	16	24	32	4					

loop ulta chalengy! → Ku? → same rahega



72 → Not prime ise maaf kredia
31 → Yeh prime hai isse remove krengy → isse (72) ka index (5) hogaya
19 → yeh prime hai → remove → isse (72) ka index (4) hogaya
14 → Not prime isse maafi dedi
3 → prime → remove → isse (14) ka index (2) par ayega and (72) ka index (1) hoga!
18 → Not prime maaf
7 → prime → remove → isse (18) ka index (0)
(14) ka index (1)
(72) —— (2)

```
P C m {
P s v solution (Arraylist <Integer> al) {
 for (int i = al.size()-1; i >= 0 ; i--) {
     int val = al.get (i);
      if (isPrime(val) == true) {
          al.remove (i);
      }
     }
}

P s boolean isPrime (int val) {
    for (int div = 2 ; div * div <= val ; div++) {
        if (val % div == 0) {
            return false;
        }
    }
    return true;
}

P s v m (S [] a) {
Scanner s = new Scanner (System.in);
int n = s.nextInt();
Arraylist <Integer> al = new Arraylist <>();
 for (int i=0; i<n ; i++) {
    al.add (s.nextInt());
 }
 Solution (al);
 System.out.println (al);
}
}
```

Time Complexity : $O(n^2)$ ← traversing the arraylist removing the prime no.] $O(n)$
internal shifting of the element] $O(n)$

Space Complexity : $O(1)$