

Day 40

DSA

13 Oct, 2021

String continued.....

Page No.	
Date	

DSA Strings Agenda for Today

1. String Interning & Immutability → done
2. Introduction to String Builder & purpose → done
3. Toggle Case → done
4. String with differences of chars → hw
5. String permutations → Test / hw
6. Introduction to ArrayList → Pending
7. Remove Duplicates → Pending

1. String Interning & Immutability → Explained

briefly in Day 36 Notes

String Interning

What?

Searching for an identical string literal in the Intern Pool area, if there is same literal present, then share same memory.

Why?

It's done to save memory space

Implications

We should never use "==" for comparison of strings, use (equals) method because it will check the correct address of string.

Strings became Immutable

Implications

Performance issues, अंगार हम से कि String की data change किए जाएँगे, लेकिन new String की जगह old, in result Time & Space Complexity increases.

Solution

String Builder solves all these problems of String.

What?

Reference are mutable, Instances are not.

Why?

Because of Interning many string points to same instance, & changing 1 instance would change many strings (to avoid that, we are not allowed to any instance by JAVA)

Implications

Some functions are missing like we are not able to change the instance

Reference होता है जिसके address पढ़ा है।
Instance होता है।

Page No.

Date

String Immutability & Implementation

Either modification function for Instance
(data of String) is not available
(like setcharAt, remove) → Not available

Or If there is Modification function
available for Instance of String,
then its performance is Poor

because it makes a complete new
copy of entire string & then makes
change. It increases the Space &
Time and Space Complexity increases
as a result.

Time Complexity becomes $O(n^2)$ instead
we expect it to be $O(n)$

STRING में Modification करने के लिए यह function
होता है → **Replace**

String s = "Hello";

~~s.replace('h', 'b');~~

System.out.println(s);

Output → **Hello**

इससे

Hello

दृष्टि point

इसी समान कार्यक्रम से

variable में address

तो पहले वाली String

की ओर Saved होगी

जो उसकी address होगी

S = s.replace('h', 'b');

System.out.println(s);

Output → **bello**

New String
जी की

replace

करने

पर

उसकी

address preference

Same करते हैं S variable में तो वह changed string print होगी।

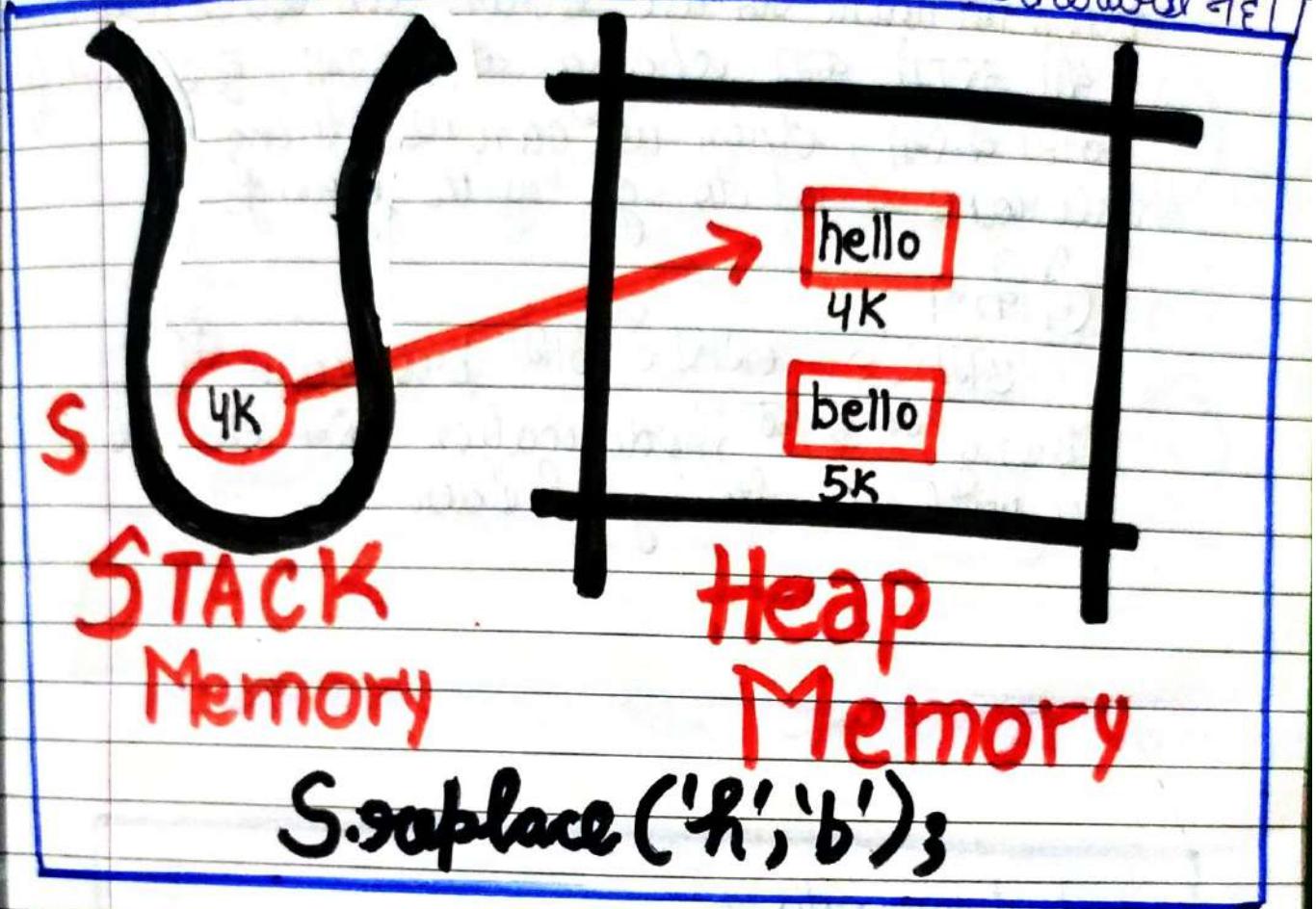
Replace करके receive करना
is most important part here.

Page No.	
Date	

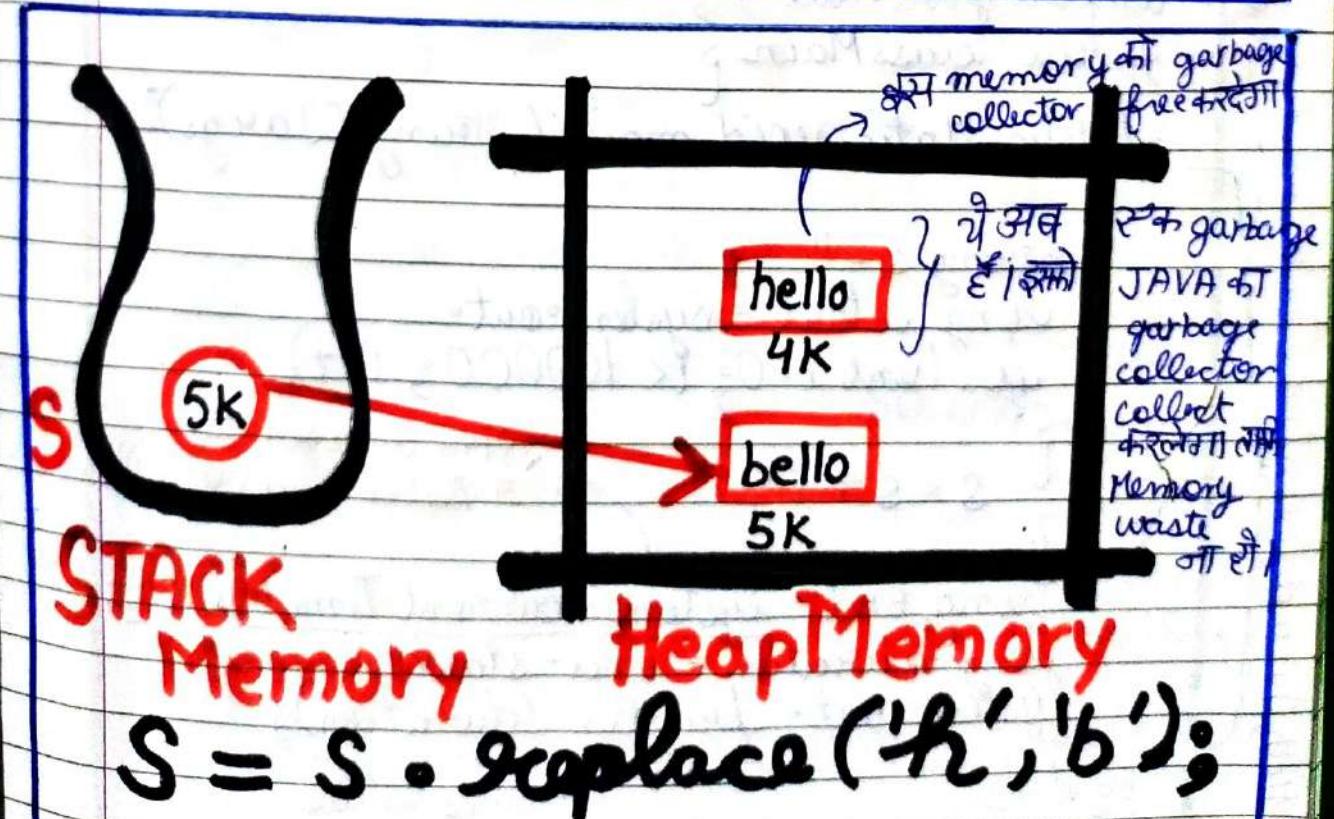
$S = S.replace('h', 'b');$

जो बदलता है

Otherwise नहीं



`S.replace('h', 'b');`



`S = S.replace('h', 'b');`

StringBuilder is the solution to String.
String is not meant for modifications.
When we want to use String in such a form
की उसमें कोई change हो जाए, तो modify
तो हो जाए, then we can use String.
We are good to go with strings.

ਪੰਜਾਬ

किन अगर इसमें लगाए दृष्टि आगे program में

If we want to modify the string, then we will use **String Builder**.

Proof → Performance of String. is Poor } Both Space & Time Complexity

Program to modify / change value / data of String 1L times

```

Time Limit Exceeded
import java.util.*;
public class Main {
    public static void main(String[] args) {
        String s = "";
        long start = System.currentTimeMillis();
        for (int i = 0; i < 100000; i++) {
            s = s + i;
        }
        long end = System.currentTimeMillis();
        long duration = end - start;
        System.out.println(duration);
    }
}

```

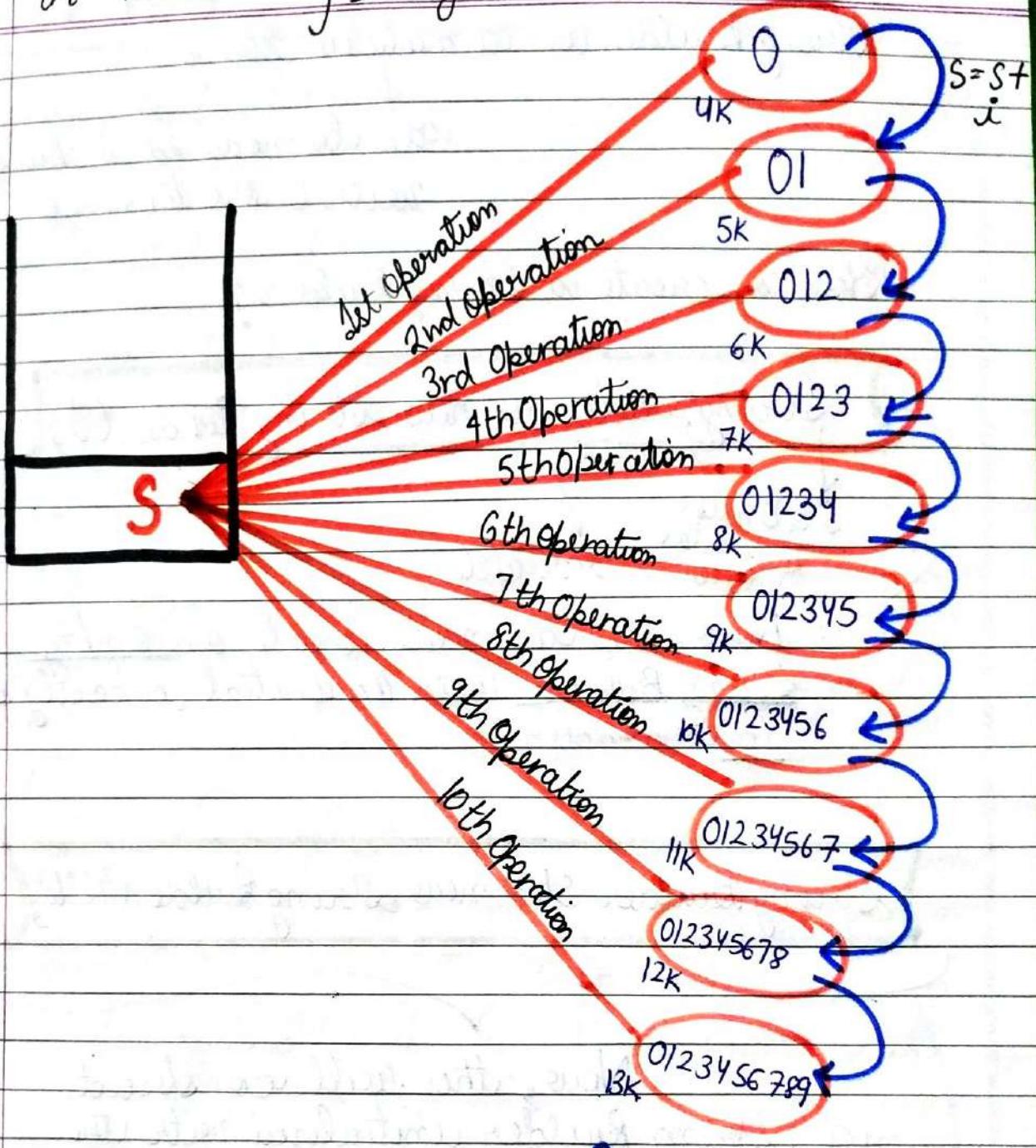
~~Java Script~~
~~of Date.now() का लिए~~

Time Limit Exceeded

Time Limit Exceeded

Each Modification on String Is taking
 Ω^2 Time Complexity...

Page No.	
Date	



& so on...

So Time Taken by 1 String =

→ Time of 1st Op + Time of 2nd Operation + ————— Time of nth operation

→ $1+2+3+4+5+\dots+n$

→ $\frac{n(n+1)}{2}$ = Time Complexity = n^2 } \therefore 1 Modification/
1 Loop on

String is Very Harmful

STRING BUILDER

→ String Builder is a mutable string

↓
can be modified & changed
with 0 shortcomings

How to create a String Builder?

StringBuilder sb = new StringBuilder();

S & B must
be capital
letter

variables

Now, this line will create an empty
StringBuilder with an initial capacity of
16 characters.

StringBuilder sb = new StringBuilder("Hello");

Now, this will construct
a String Builder initialized with the
contents ("Hello").

Program to modify/change value of String Builder 1L Times

```
→ import java.util.*;  
→ public class Main {
```

```
public static void main (String[] args)  
{
```

```
    long start = System.currentTimeMillis();
```

long start is digit of timestamp
gives milliseconds
passed since
1 Jan 1970

```
// String s = "";
```

```
// for (int i=0; i<10000; i++)
```

```
// {  
//     s=s+l;
```

```
// }
```

10000^2 operations = 10^8 operations

for string, if we
run this only
for 10K times,
the string will
take lots &
lots of time.

```
// long end = System.currentTimeMillis();
```

```
// System.out.println(duration);
```

commented

Output = 145 Milliseconds

```
StringBuilder sb = new StringBuilder();
```

```
for (int i=0; i<10000; i++)
```

```
{
```

```
    sb.append(l);
```

```
}
```

Use append
func
here to add

```
long end = System.currentTimeMillis();
```

```
long duration = end - start;
```

```
System.out.println(duration);
```

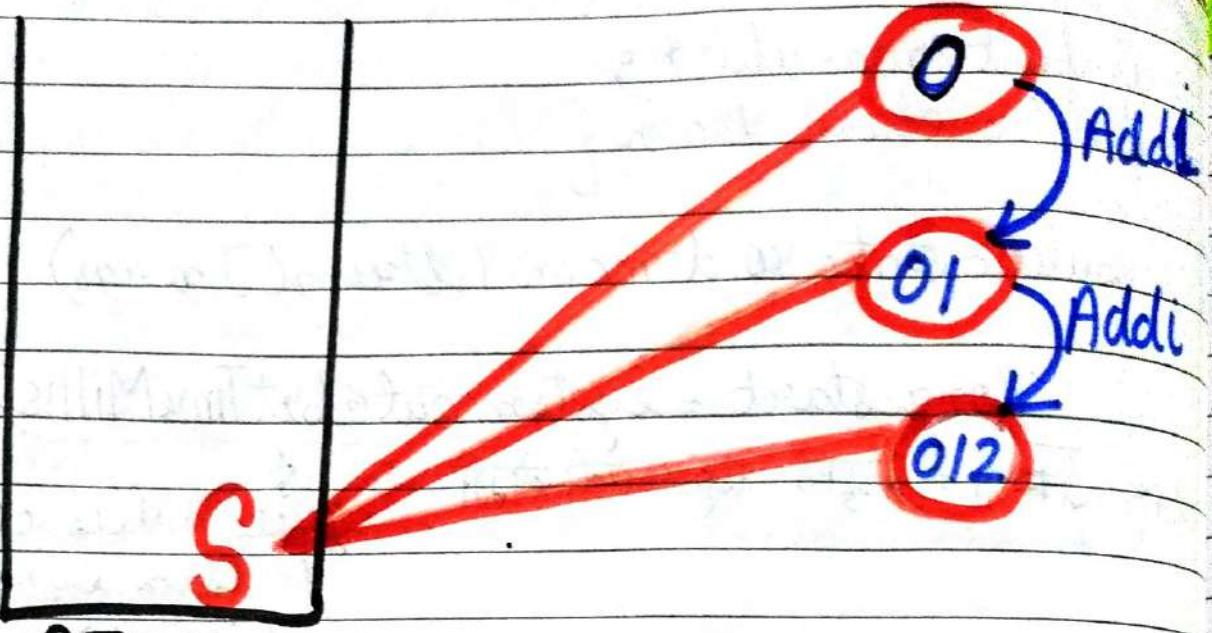
```
}
```

Output = 10 Milliseconds

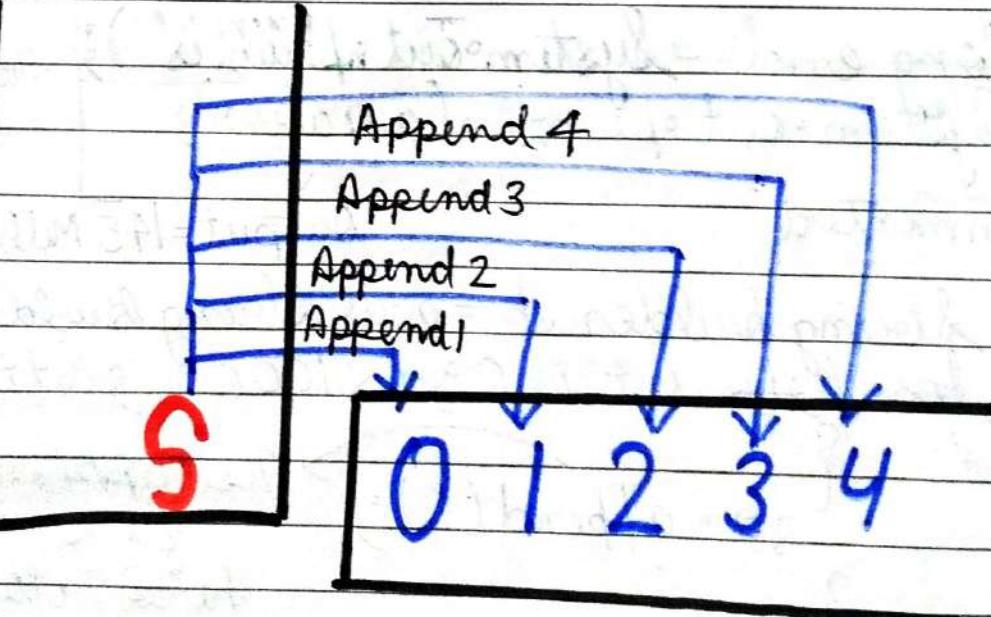
HUGE TIME DIFFERENCE

USING STRING

STACK
MEMORY



USING STRINGBUILDER



Methods of STRINGBUILDER

StringBuilder sb = new StringBuilder("Hello");

1. sb.length() → if parenthesis is removed, string, stringbuilder would return the length (character count) of String Builder

array of length () जैसी कार्यतात्विकी
length () जैसी कार्यतात्विकी

2. sb.charAt (int index)

returns the character at specified Index.

→ Capital

3. sb.setCharAt (int index, char ch);

if Camel Case follows

Sets (Over writes) the character at specified index as content of ch.

→ It function String setCharAt (setcharAt) of String Builder provides

4. sb.append (Object obj) **append is O(1)**

Appends the parameter passed as string.
The object can range from datatype

→ int

→ float

→ string

→ character

→ Sequence of character & anything

→ Capital

5. sb.deleteCharAt (int idx)

Camel Case

Deletes a character at the specified index.

6. `sb.replace(int startidx, int end_idx,
String str)`

Replaces the substring contained
between start and end index
with the specified string str.

7. `sb.delete (int start_idx, int end_idx)`

Deletes the substring contained
between the start and end index.

8. `sb.toString()`

returns a string representing
the contents of sb (String Builder Object).

StringBuilder :- Methods

<code>

```
import java.util.*;  
public class Main {  
    public static void main (String [] args) {  
        StringBuilde sb = new StringBuilde ("Hello");  
        // Printing sb StringBuilde  
        System.out.println ("String : " + sb);  
        → Output → String : Hello
```

1. // Printing the length of sb StringBuilde

Length
System.out.println ("Length of String : "
+ sb.length());
→ Output → Length of String : 5

2. // Printing the first character of sb StringBuilde

charAt
System.out.println ("First Character : "
+ sb.charAt(0));
→ Output → First Character : H

3. // Setting the first character as 'h'

setCharAt (0, 'h');
↳ small letter
System.out.println ("String : " + sb);
→ Output → String : hello
↳ first character of 'h'
set कर दिया, पढ़ो 'H'
2T

4. // Appending the string 'reader' to
sb String Builder.

append

sb.append(" reader");

System.out.println("String : " + sb);

Output → String : hello reader.

5. // Delete the character at index 5

deleteCharAt

sb.deleteCharAt(5);

System.out.println("String : " + sb);

Output → String : hello reader,

dated the space at
5th index

→ 4th hello reader, 8th

→ 3rd hellorader &

→ String 3rd String Builder &
even spaces are counted as
a character

6. // Replace the characters / substring
b/w 5th index to 10th index with
"bella"

replace

sb.replace(5, 10, "bella");

System.out.println("String : " + sb);

Output → String : hellobella

4th to 10th reader 8th

7. // delete the substring contained
between the start & end index

sb.delete(5, 10);

System.out.println("String : "+sb);

Output : String : Hello;

Actual hello**bello** eff

8. // Insert a character at 5th index

Eg) Insert o at 5th index

sb.insert(5, 'O');

System.out.println("String : "+sb);

Output → String : helloo

↑
inserted o
at 5th index

Eg) Insert 'l' at 0th index.

sb.insert(0, 'l');

System.out.println("String : "+sb);

Output → String : lhelloo

Eg) Insert 'l' at 4th index

sb.insert(4, 'l');

System.out.println("String : "+sb);

Output → String : ihelloo

↳ 4th index

Difference between setCharAt & insert sb="hello"

setCharAt

```
sb.setCharAt(1, 'd');  
System.out.println(sb);
```

Output → hdollo

setCharAt तक

index पर जो

character है

उसकी गेम character

से replace करवेगा।

insert

```
sb.insert(1, 'd');  
System.out.println(sb);
```

Output → hdello

insert तक

index पर

character

insert

पढ़वेगा, वही उस index

और आगे वाले सारे

character 1 + index

आगे shift हो जायगा।

Ques. 1

TOGGLE CASE

- Q. - We are given a string and we are expected to change the case of each character from lower case to upper case or, if, upper case, then its lower case.

Eg.) pepCODING → हमें सभी lower case letters को upper case letter का change करना है और सभी upper case characters/letters को lower case letters का

↓
PEPcod INg

* हर character का सक ASCII code होता है।	
* 'A' का ASCII code 65 होता है।	65-90
* 'B' का ASCII code 66 होता है।	
* 'C' का ASCII code 67 होता है।	26
* 'D' का ASCII code 68 होता है।	Capital
* 'E' का ASCII code 69 होता है।	letter
; & so on... 'Z' का ASCII code 90 होता है।	

* 'a' का ASCII code 97 होता है।	97-122
* 'b' का ASCII code 98 होता है।	
* 'c' का ASCII code 99 होता है।	
* 'd' का ASCII code 100 होता है।	
* 'e' का ASCII code 101 होता है।	

& so on

'z' का ASCII code 122 होता है

A को a में convert करने के लिए यह formula
आविष्ट।
a से A convert करने के लिए यही यह formula
आविष्ट।

$$E^2 - A^2 = 4 \text{ at difference } 4\%$$

Similarly ,

$$6e^9 - 6a^9 = 4$$

तो हम कह सकते हैं कि दोनों equal हैं

$$|E'-A'| = |e'-a'|$$

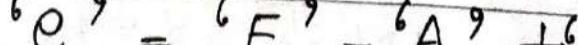
Now, after jiggling with this equation,
we get

Converting lowercase to uppercase

$$E' = e' - a' + A'$$

Converting uppercase to lowercase

$$'c' = 'E' - 'A' + 'a'$$


 Arrows point from the lowercase letters 'c', 'a', and 'e' to their corresponding uppercase letters 'C', 'A', and 'E' in the equation.

$$\text{UC} = \text{LC} - 'a' + 'A'$$

Uppercase

$$\text{LC} = \text{UP} - 'A' + 'a'$$

Lowercase

STEPS TO CODE:-

Step 1) Take the input string and pass the input string to ToggleCase function

Name of function that we will create is Toggle Case.

Step 2) Since, we need to toggle (make changes) to the string, we will use a String Builder object.

We have to toggle each character, thus, we will traverse through every character using a loop (from index 0 till the last index).

Step 3) When we reach at some index ; lets say 3, we store the character at index 3 in ch. Now, we have the character to toggle & we will now check, if it is an uppercase character or a lowercase character.

If the character is between A and Z (both inclusive), then its surely an uppercase character, otherwise we will check if its a lowercase letter using a similar condition.

Step 4) Let's take an example:-

Let $ch = "f"$

Then the else if condition would become true.

Using, $\boxed{\text{char uc} = (\text{char})(\text{ch} - 'a' + 'A')}$,
uppercase.

we will convert " f " to " F "

$$UC = LC - 'a' + 'A'$$

↑
 f

$$= ('f' - 'a') + 'A'$$

$$= (102 - 97) + 65$$

$$= 5 + 65$$

$$= 70$$

↳ returns an integer which needed to be converted to char using explicit conversion.

Since, the calculation results in an integer corresponding to the required uppercase character (Type Casting - explicit type casting)

Step 5) Now, using setCharAt() function, we overwrite the character present at this index i with the character in the opposite case we managed to find.

if result आया तो ३स्को (char)(result)

explicitly Typecast नहीं

setCharAt of ith index QR

overwrite करो।

Step 6) Now, return the new toggled string back to main().

Toggle case <code>

```
→ import java.io.*;  
→ import java.util.*;  
→ public class Main{
```

```
    public static String toggleCase  
        (String str){  
    }
```

```
    StringBuilder sb = new StringBuilder(str);
```

Given String को ~~String~~ ~~StringBuilder~~
में convert कर ताकि इसे modification
कर पाए।

```
for (int i=0; i<sb.length(); i++) {
```

↳ ~~String~~ Builder के हर एक
character पर Traversal
करना है हमें

```
    char ch = sb.charAt(i);
```

↳ ith index पर वही एक
character है उसको ch में
store करता है।

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

→ अब char
'A' से 'Z'

→ it explicit Type Casting कर रहा है। क्योंकि

```
    ch = (char) (ch + 'a' - 'A');
```

मिला है

converting
to
lower
case

उसे
lowercase
में convert
करते

else
{
}

अगर हमने else condition की entry किया है तो इसका मानक character lowercase character से और अपने uppercase में हो convert करना है।

$$ch = (\text{char})(ch + 'A' - 'a');$$

Converting to
upper case

sb = SetCharAt(i, ch);

↳ 37th i th Index का char
का changed character को
overwrite करता है।

{
}

return sb.toString();

↳ String Builder से वापस
String में convert होता है
return होता है।

public static void main (String [] args)

↳ It Main function

→ Scanner scnr = new Scanner
(System.in);

→ String str = scnr.next(); → String का Input
मात्र

→ System.out.println(toggleCase(str));

Toggle function का call होता है।

{
}

14 Oct 2021

DAY STRING CONTINUED.....

41

Agenda

1. String Permutations → HW
2. String with difference of chars.
3. Introduction to ArrayLists → CW
4. Remove Primes.
5. Print Decreasing
6. Print Decreasing Increasing
7. Factorial
8. Power Linear.
9. Power Logarithmic

Ques. 1

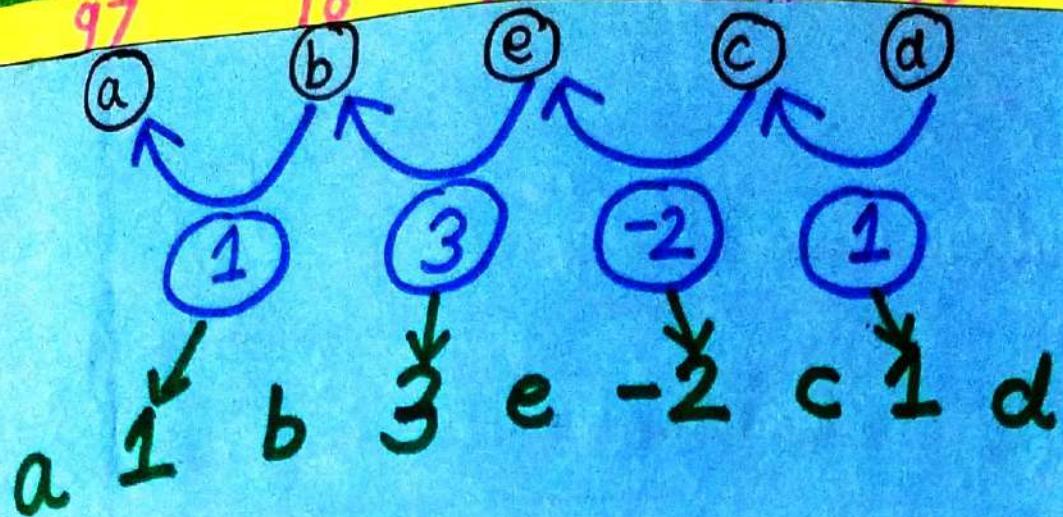
STRING WITH DIFFERENCE OF EVERY TWO CONSECUTIVE CHARACTER

→ We are given a string that contains only lowercase & uppercase alphabets.

→ एक स्ट्रिंग दिया गया है जो कि उसमें अक्षरों के ASCII values का अंतर है।

Eg → For abecd

Answer → a1b3c-2d1e

Approach

1. We will use a `StringBuilder` and initially append the first character of the string.
2. Run a for loop from 1 to `str.length() - 1` and at each index, find the difference between the current character and the previous character.
3. Append the difference & then append the current character.

The reason why we will be using a `StringBuilder` is that it is much easier to operate on `StringBuilder` as compared to `String`.

To get the difference in the ASCII value, we can just simply subtract the characters. Because implicitly all the characters are represented by their ASCII value, so while subtracting it is their ASCII value that gets subtracted.

Finally, we will append both the difference and the current char to maintain the form the output is expected.

→ import java.*;
→ import java.util.*;
public class Main {

public static String solution (String str)

String is the return type
Page No. _____
Date _____

Page No.

Date

StringBuilder sb = new StringBuilder();

L, यह string builder का प्रिंटर है।

sb.append(str.charAt(0));

यह sb में सबसे पहली 0th index का char store करता है।

for (int i = 1; i < str.length(); i++)

1 index से str.length() - 1

index तक loop चलता है।

char curr = str.charAt(i); → current element का

char prev = str.charAt(i - 1); → उसका previous element का

int gap = curr - prev;

store करता है।
curr का prev variable है।

sb.append(gap);

current char
का ASCII value

prev का

subtract

फिर save the

difference

Remember

that characters

are represented

by their
ASCII values

Now append
this difference
in the StringBuilder

sb.append(curr);

Now append the current
character in the StringBuilder

→ return sb.toString();

↳ String Builder ↗

~~if string is convert from
file return that~~

→ public static void main (String [args])

{ Scanner scn = new Scanner (System.in);

String str = scn.next();

System.out.println (solution (str));

* Difference b/w append & insert

append ↓

at last

index ↗

at add ↗

insert ↗ at

index ↗ add ↗

↗

Introduction to ArrayList

it's a class

* अब हम ArrayList class को use करना पड़ता है।

अगर हम एक array बनाकिया 5 size का

`int [] arr = new int[5];`

तो अब वह array size 5 का ही रहेगा।

लेकिन ArrayList में ऐसे सकता है, कम हो सकता है size

हम जितना चाहे उन्हीं size का जावा कर सकते हैं ArrayList का

शुरूआत में तो size 0 रहता है, अब हम इसमें जितनी भी size की value डालना चाहे, हम इसकर सकते हैं।

`ArrayList<Integer> list = new ArrayList<>();`

प्रियो आ
datatype
का
ArrayList
बना सकते हैं।

Name
of

Variable of ArrayList Type

मनो ये declare किया है

ArrayList को।

int नहीं लिखता बदा Integer ही लिखता होता है with capital I. (small int fast चलता है, Capital Integer तो slow चलता है)

Performance array प्रैसी ही देगा, बस किसी में array प्रैसा नहीं है।

① → `ArrayList<Integer> list = new ArrayList<>();`
 → `System.out.println(list + " -> " + list.size());`

↑ अगरी तो empty
Page No. _____ Date _____

→ **ArrayList का size कैसे पढ़ता है?**
Length of string printed
ArrayList के Parathesis में से एक चाहिए।
Size होता है 0
ही होता है

→ **Output →** `[] -> 0`

② → **ArrayList के content-add करने के लिए func होता है add()**

→ `list.add(10);` → **ArrayList के add करना**
 → `list.add(20);` → **it values.**
 → `list.add(30);`
 → `System.out.println(list + " -> " + list.size());`

→ **Output →** `[10, 20, 30] -> 3`

③ → **ArrayList के insert करने के लिए func होता है add(index, value)**
 → `list.add(1, 1000);`
 → `System.out.println(list + " -> " + list.size());`

→ **Output →** `[10, 1000, 20, 30] -> 4`

(4) → ArrayList से कोहै गी value get करनी है
तो उसके लिए भी function होता है
`get(index);`

```
int val = list.get(1); // not list[1]  
System.out.println(val);
```

→ Output → 1000

array में list[1]
लिस्ट में लेकिन गए
get func use
बड़ा पड़ा॥ हमे

(5) → ArrayList में value set करने के लिए भी Set
function होता है।

`set(index, value);`

```
list.set(1, 2000); // not list[1]=2000
```

```
System.out.println(list);
```

→ Output → [10, 2000, 20, 30] → 4

(6) → Remove एक ArrayList का एक function होता है;
इसमें किसी गी index पर value हटा सकते हैं।

`list.remove(1);`

```
System.out.println(list + " -> " + list.size());
```

→ Output → [10, 20, 30] → 3

(7) → ArrayList में strings की भी छाना सकते हैं।

```
ArrayList<String> l2 = new ArrayList<String>;  
l2.add("Hello");  
l2.add("Bello");  
l2.add("Cello");  
System.out.println(l2 + " -> " + l2.size());
```

→ Output → [Hello, Bello, Cello] -> 3

⑧ → ArrayList में एट लॉप से जाना सकते हैं।

Way 1. → if for loop है।

Page No. _____

```
for (int i=0; i<list.size(); i++)  
{    int val1 = list.get(i);  
    System.out.println(val1);}
```

1. loop जितना लॉप है।
2. ArrayList
3. इन्डेक्स
4. कोई वैल्यू
5. की वैल्यू
6. get करता है।
7. print करता है।

3. Displaying an ArrayList

Way 2 → Using for each loop

→ for each integer i in the list, print i

```
for (int i : list)
```

```
{    System.out.println(i);}
```

3

ArrayList in brief.....

1. ArrayList is a data-structure (just like array but very different).
2. In Java, Developers have written a class for ArrayList and its methods. Therefore,
∴ It is ^{also} called "ArrayList class"

If you don't know
what is class
don't worry, इसके
बारे में बाद में OOPS
में पढ़ींगी।

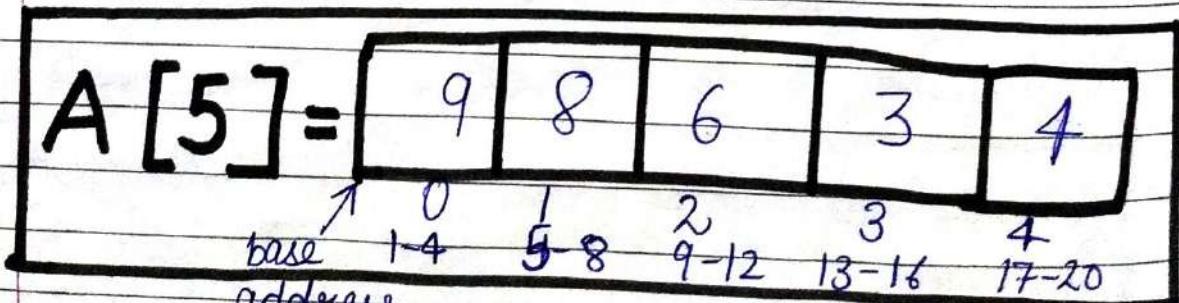
3. What is ArrayList ?

इससे पहले discuss करते हैं।

What is ARRAY ?

An Array is a data-structure, which is a collection of similar data types which are arranged in a contiguous memory.

जटिल नहीं बिल्कुल



of A is 201.

जैसे यह Array ग्रीन के size 5 का बना है, उसमें इसके अलावा भी यहाँ elements भी दी जिन्हें इसके बाहर store होते हैं (Therefore, we can say that array has contiguous memory).

Page No.	
Date	

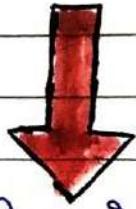
इस Array की Base Address 201 है।
और हर Integer 4 byte की होती है।

∴ एक element 4 byte of memory space की
consume करता है।

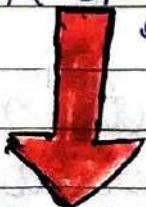
∴ The 1st element present at Index 0
starts at memory address 201 & ends at
memory address 204.

The Second element at Index 1
starts at 205 memory address without
any gap of even a single byte in
between 2 index.

This happens in the entire array & so,
the memory is Contiguous.



ये सब ती दो पदों से पता है,
परसे क्यों पढ़ते हैं Array की Memory Allocation
के बारे में?



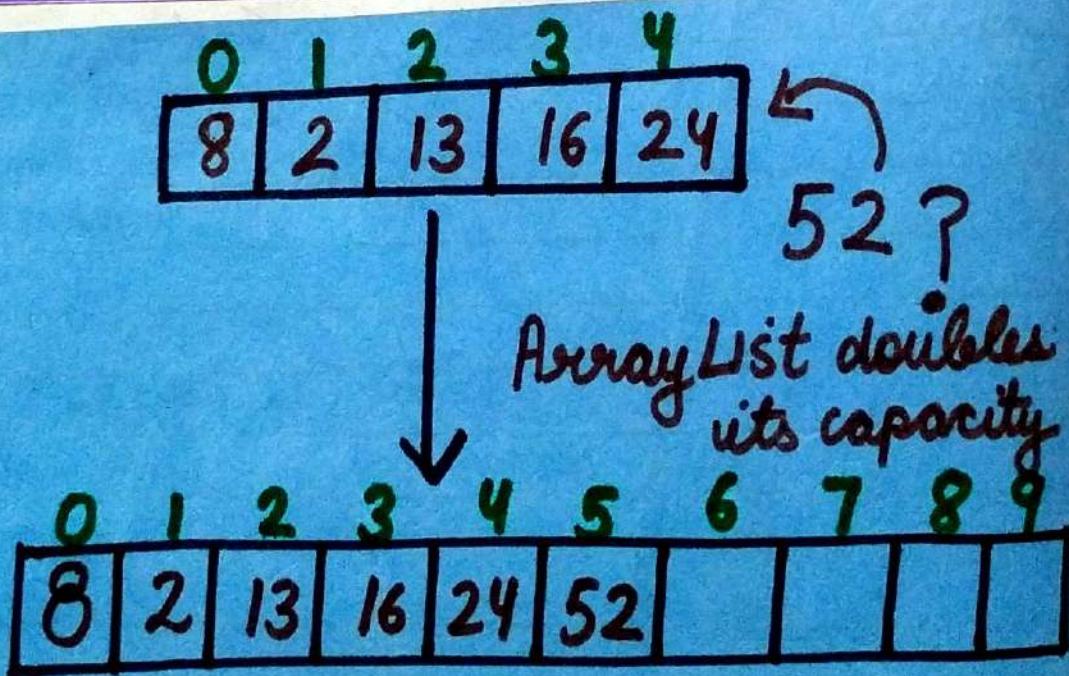
फिर ऐसे ArrayList also follow these rules

* ArrayList will also have a contiguous memory, but there is a difference

Page No. _____
Date _____

We need to define the size of array for creating an array in the memory. But we do not define the size of an ArrayList. This is the major difference between Array and ArrayList.

4. How an ArrayList work?



So, there is something called as Size and Capacity

Size of an Array/ArrayList is the number of elements that are currently present in it.

Page No. _____
Date _____

Capacity is the total number of elements that can be present in the Array / ArrayList.

→ Size of the above ArrayList was 5 & the Capacity was also 5. So the ArrayList was complete.

अब अब इसे तो संपूर्ण कहा दें।

ArrayList में 52 add करोगा,
तो अब कहाँ होगा ?

→ जैसा कि पहले भी बताया जा चुका है

ArrayList will manage the size itself.
लैकिन कहाँ ?

ArrayList actually doubles the total capacity of the list.

So, the current capacity was 5.
When 52 was inserted, the capacity became 10 and after the insertion of 52, there were still 4 vacant spaces for insertion.

8	2	13	16	24	52	45	36	54	92
---	---	----	----	----	----	----	----	----	----

7?

Doubled the Capacity

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
8	2	13	16	24	52	45	36	54	92	7									

In previous ArrayList, after 52, we inserted 43, 36, 54, 92, and the ArrayList did not expand as the capacity was enough to store all those elements. When we entered 7, the total capacity doubled again & became 20. Similarly, the capacity will not expand on entering 9 more elements.

The initial size of ArrayList is 0 as it is empty and the initial capacity of an ArrayList in Java is **10**

Ques. 2).

REMOVE PRIMES

Page No. _____
Date. _____

→ We have to remove all prime numbers from an ArrayList.

Eg 1)

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

After Removing Primes

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

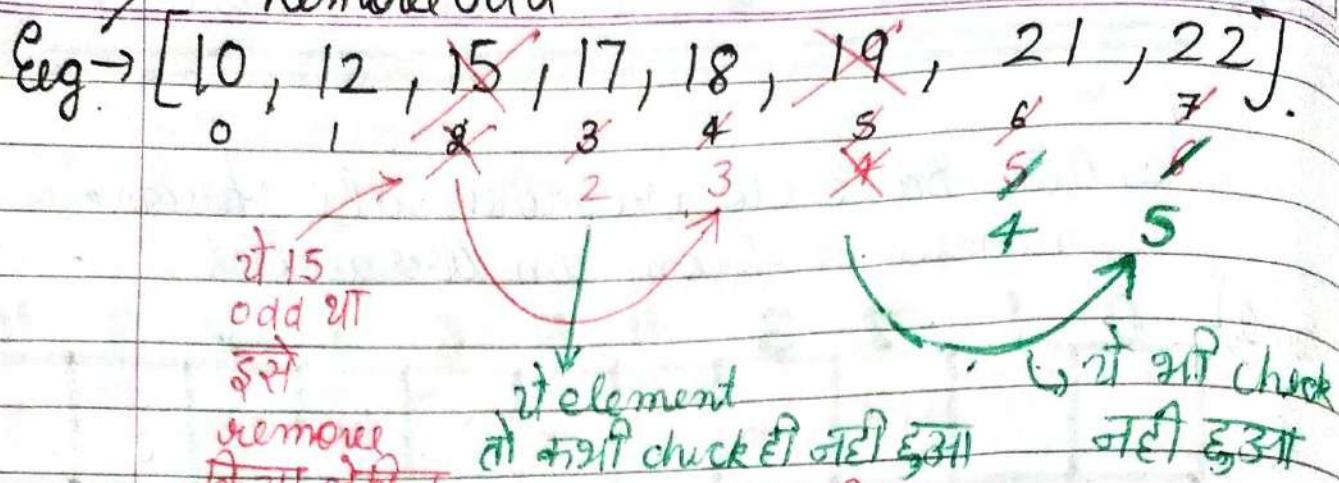
→ This is an ArrayList with both prime & non-prime no. As well. We have to remove all the prime no. from the ArrayList.

→ This should be the ArrayList we get after removing prime no.

Approach

→ ArrayList के element को remove
करने के लिए कैसी गति Loop पर्हें
नहीं होला, Loop की समाप्ति तक
पर्हें है।

To Remove Odd



a) पूर्वी ही 15 को remove किया

पूर्वी element 3, 4, 5, 6, 7 index पर थे,
तो अब इन index decrease होते हैं और
2, 3, 4, 5, 6 पर आते हैं।

b) अब जैसे ही 15 remove हुआ (++ हुआ),

अब i की value होगी 2 से 3 तक

और अब 3rd index से loop

continues होगा, 2nd index तक element

17 तो कमी check ही नहीं हुआ तो loop

पुरा traversal करने के बाद भी answer

तो गलत ही आएगा, because हम पूरा

traversal करते हैं।

∴ ArrayList में element को remove

करने के लिए loop की हमेशा उल्टा

Traversal करते ही traversal करते

गलत answer कमी सही नहीं आएगा।

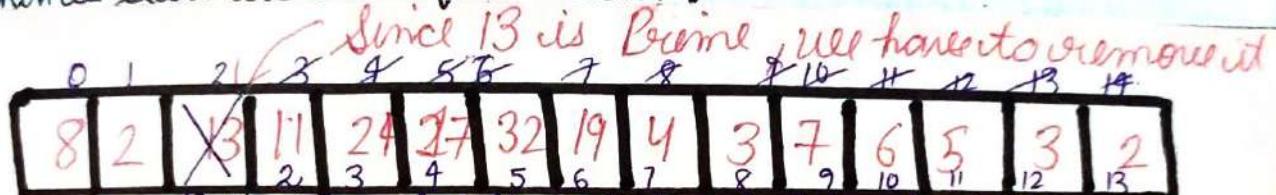
Pseudocode :-

1) Declare a variable 'i' and keep it at the end of ArrayList i.e. the index at which it has its last value.

* Also, write the function for checking whether a number is prime or not.

2) Iterate through the ArrayList by decrementing the 'i' variable and keep on removing the prime numbers by checking the numbers with the help of your creation function.

When we start iteration from FRONT ↴



When we remove 13, the internal shifting of elements takes place & the index 11 has now 11, 11 is at Index 2 & it is also a prime no., but if we look at code carefully, we simply increment the value of 'i' variable, so it became 3. This means, that in the next iteration, we will check whether the element at index 3 is prime or not, but we had a prime number at index 2 that we already missed. This causes a problem & due to this, all the prime no. might not get removed from the ArrayList.

How can we solve this issue? This issue can be solved by starting the iteration from back of the ArrayList.

When we start iteration from REAR ↴



The 'i' variable started from index 10. At index 10 it found 4, which is not a prime no. So, it moved backward at index 9 where it found 7, which is a prime no. So, 7 got removed & 4 took its place/index and the index of 3 did not change. So, next time when 'i' will be decremented & it checks for index 8 & find out that value 3 is also prime, 3 will also get removed. So when we iterate the ArrayList in reverse direction, there will be no issue.

ARRAYLIST - REMOVE PRIMES

WAY 1

REAR ITERATION

<CODE>

Iteration Start from
END

```
→ import java.util.*;  
→ import java.io.*;  
→ public class Main {
```

```
public static void solution (ArrayList<Integer>  
al) {
```

```
for (int i = al.size(); i >= 0; i--)
```

}

ArrayList → reverse
order → traverse

if (isPrime (al.get (i))).

isPrime
function

{ al.remove (i); } ← array
list → ER

} ← element
→ Pass

```
public static boolean isPrime (int n)
```

{

```
for (int i = 2; i * i <= n; i++)
```

→ element

```
if (n % i == 0)
```

→ prime

```
return false;
```

→ if

```
}
```

→ if

```
return true;
```

→ if

```
}
```

→ if

public static void main (String args)

<i>Page No.</i>	
<i>Date</i>	

```
{ Scanner scon = new Scanner(System.in),  
    int n = scon.nextInt();  
    ArrayList < Integer > al = new ArrayList();  
    for (int i = 0; i < n; i++) {  
        al.add(scon.nextInt());  
    }  
    solution(al);  
    System.out.println(al);
```

 ~~ABR~~ में front traverse करने की ही condition
जैसा कि हमारे पास यह prime element
को remove करेंगे, साथ में i-th तिकड़ा
यह तिकड़ा element remove के next, तो
वह just next ith element को भी check
कर सके।

WAY2 → REMOVE ALL PRIMES <code> ITERATION START FROM FRONT

```
→ import java.util.*;  
→ import java.io.*;  
→ public class Main  
    public static void main( String [ ] args )  
    { Scanner scan = new Scanner ( System . in );  
        int n = scan . nextInt ();  
        ArrayList < Integer > al = new ArrayList < > ();  
        for ( int i = 0 ; i < n ; i ++ )  
        { al . add ( scan . nextInt () );  
            solution ( al );  
        System . out . println ( al );  
    }
```

```
→ public static void solution ( ArrayList < Integer > al )  
    { for ( int i = 0 ; i < al . size () ; i ++ )  
        { int val = al . get ( i );  
            boolean isThisValPrime = isPrime ( val );  
            if ( isThisValPrime == true )  
                { al . remove ( i );  
                    i -- ;  
                }  
        }
```

] Most Important Step

```
→ public static boolean isThisValPrime ( int n )  
    { for ( int i = 2 ; i * i <= n ; i ++ )  
        { if ( n % i == false )  
            { return false ; }  
        }  
    return true ;  
}
```

HW

Page No.	
Date	

Ques. 01)

Print All Permutations of a String Iteratively

Permutations of "abc" are :-

1. abc
2. bac
3. cab
4. acb
5. bca
6. cba

"abc" has 6 permutations

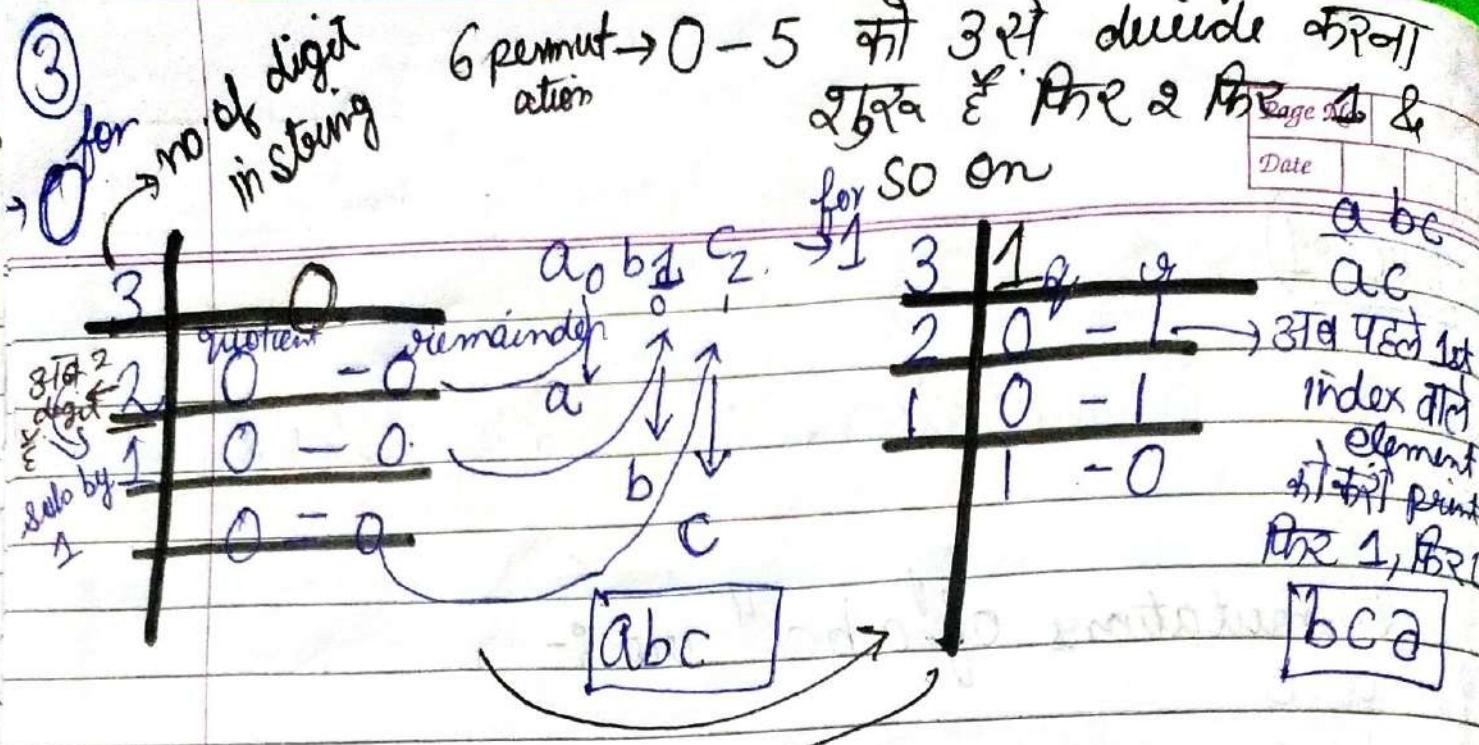
A general formula for permutations is $(n \text{ factorial})$ where n is the length of the string.

* Assumption :- We are assuming that all characters are unique.

Approach.

1. सबसे पहले string की length check करो।
= 3.
अतः इसमें no. of permutation होंगी = 3
 $= 3 \times 2 \times 1$
 $= 6.$

2. तो 6 permutations हैं।



for 2

3	2	2
2	0	2
1	1	0
1	0	

abc \rightarrow ab \rightarrow b for 3

3	3	3
2	1	0
1	0	1
0	0	

for 4

3	4
2	1 - pr
1	0 - 1
0	0 - 0

for 5

3	5
2	1 - 2
1	0 - 1
0	0 - 0

cba.

Approach :-

1. We will iterate through 0 to $(n-1)$
2. For each number, divide it by n , then by $n-1$ & then by $n-2$ & so on... | factorial n is represented as f & l_n
3. Whatever comes as the remainder in each case, we will remove that char from the string, print it & use the quotient for next division purpose (we will not make any direct changes to the original string, we will make changes to a copy).

& This way, for every iteration 0 to $n-1$, we will get one string of length n .

& Interestingly all these strings will be unique permutation of given string

$$3 \quad 0 \quad \begin{array}{|c|c|c|} \hline \textcircled{1} & \textcircled{2} \\ \hline \text{X} & \text{bc} \\ \hline \end{array} \Rightarrow a.$$

$$2 \quad 0-0 \quad \begin{array}{|c|c|c|} \hline \textcircled{1} & \textcircled{2} \\ \hline \text{X} & \text{bc} \\ \hline \end{array} \Rightarrow b$$

$$1 \quad 0-0 \quad \begin{array}{|c|c|c|} \hline \textcircled{1} & \textcircled{2} \\ \hline \text{X} & \text{bc} \\ \hline \end{array} \Rightarrow c$$

abc

for 0

↳ 0 is being divided

$$3 \quad 4 \quad \begin{array}{|c|c|c|} \hline \textcircled{1} & \textcircled{2} \\ \hline \text{a} & \text{X} & \text{c} \\ \hline \end{array}$$

$$2 \quad 1-1 \quad \begin{array}{|c|c|c|} \hline \textcircled{1} & \textcircled{2} \\ \hline \text{a} & \text{X} & \text{X} \\ \hline \end{array}$$

$$1 \quad 0-1 \quad \begin{array}{|c|c|c|} \hline \textcircled{1} & \textcircled{2} \\ \hline \text{a} & \text{b} & \text{c} \\ \hline \end{array}$$

for 4

↳ 4 is being divided

< code >

Page No.

Date

```
→ import java.io.*;  
→ import java.util.*;  
→ public class Main {  
    public static void main (String [ ] args)  
    {  
        }
```

```
Scanner scn = new Scanner (System  
                           .in);
```

```
String str = scn.next();  
solution (str);  
}

```
Taken
String
as
Input
```


```

factorial function

```
→ public static int factorial (int n)
```

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

```
To
calculate
the
factorial
```


```

```
→ public static void solution (String  
                             str)
```

```
{  
    int n = str.length;  
    int f = factorial (n);  
}
```

```
for(unt i = 0 ; i < f ; i++)
```

```
{ StringBuilder sb = new StringBuilder(str)
```

```
int temp = i ;
```

```
for(unt dice = n ; dice >= 1 ; dice --)
```

```
{ int q = temp / dice ;
```

```
quotient unt vr = temp % dice ;
```

```
System.out.print(sb.charAt(r));
```

```
sb.deleteCharAt(vr);
```

```
temp = q ;
```

```
System.out.println();
```

3.

~~STRING~~
**STRING
DONE**

