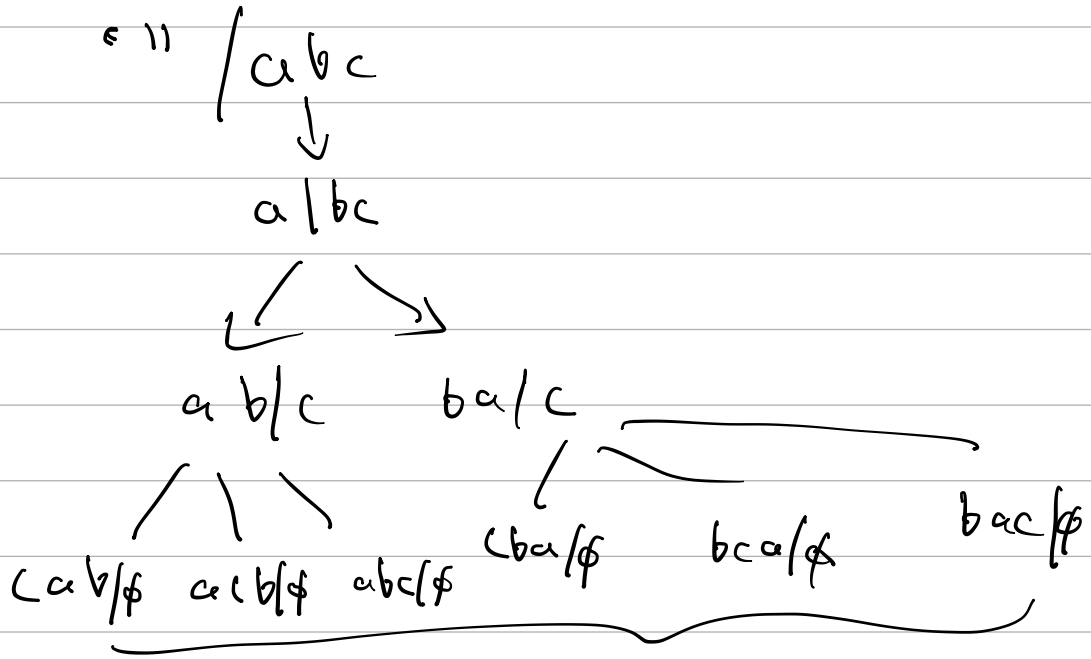
Permutations

str = abc

Permutations :- [abc, bac, cab, bca, acb, ...].



*). If length of string is N the total permutations is N!

*). We observe at each step the number of Unprocessed elements + 1

Logic :- -> Return processed when UP is empty.

-> for loop used for doing number of function calls,

-> for loop from 0 to p.length (inclusive)

P UP

"" | abc.

P.length = 0.

for loop only once.

"a" | bc

P.length = 1

for loop two time.

i = 0, 1.

-> Use two pointers f and s. to point to the location where 'ch' should be added

ab | c

In this case. -> p.length.

```
for(i=0; i<=2 ;i++)
{
String f= p.substring.(0,i);
String s= p.substring(i, p.length());
permutations(f+ch+s, up.substring(1));
}
```

skip 1st char.

*) Note:- The .substring() method do not include second argument.

Iterations:-

for i = 0.

ch = c

f = "" (Includes 0, but should not include 0 so results empty)

s = (0, 2) → ab.
(0 1)
↓
does not include 2.

Now p = f + ch + s

= cab.

for i = 1.

ch = c.

f = (0, 1) → a

s = (1, 2) → b.

p = f + ch + s → acb.

for i = 2

ch = c

f = (0, 2) → ab

s = (2, 2) → ""

p = f + ch + s → abc

→ Like this is goes on. by removing first char at UP.

CODE :-

```java
public class Permutations {
    public static void main(String[] args) {
        permutations( p: "", up: "abc");
    }

    static void permutations(String p, String up) {
        if (up.isEmpty()) {
            System.out.println(p);
            return;
        }
        char ch = up.charAt(0);
        for (int i = 0; i <= p.length(); i++) {
            String f = p.substring(0,i);
            String s = p.substring(i, p.length());
            permutations( p: f + ch + s, up.substring(1));
```

}
}
}

RETURNING THE OUTPUT AS ARRAYLIST :-

```java
    static ArrayList<String> permutationsList(String p, String up) {
        if (up.isEmpty()) {
            ArrayList<String> list = new ArrayList<>();
            list.add(p);
            return list;
        }
        char ch = up.charAt(0);

        ArrayList<String> ans = new ArrayList<>();

        for (int i = 0; i <= p.length(); i++) {
            String f = p.substring(0, i);
            String s = p.substring(i, p.length());
            ans.addAll(permutationsList( p: f + ch + s, up.substring(1)));
        }
    }
```

}.

To count the number of permutations :-

→ We get 1 permutation each time when UP becomes empty.

→ 1st Method pass a count variable as argument. Increase it in the base condition.

→ 2nd Method, local variable count.

```
static int permutationsCount(String p, String up) {
    if (up.isEmpty()) {
        return 1;
    }
    int count = 0;
    char ch = up.charAt(0);
    for (int i = 0; i <= p.length(); i++) {
        String f = p.substring(0, i);
        String s = p.substring(i, p.length());
        count = count + permutationsCount(p: f + ch + s, up.substring(1));
    }
    return count;
}
```