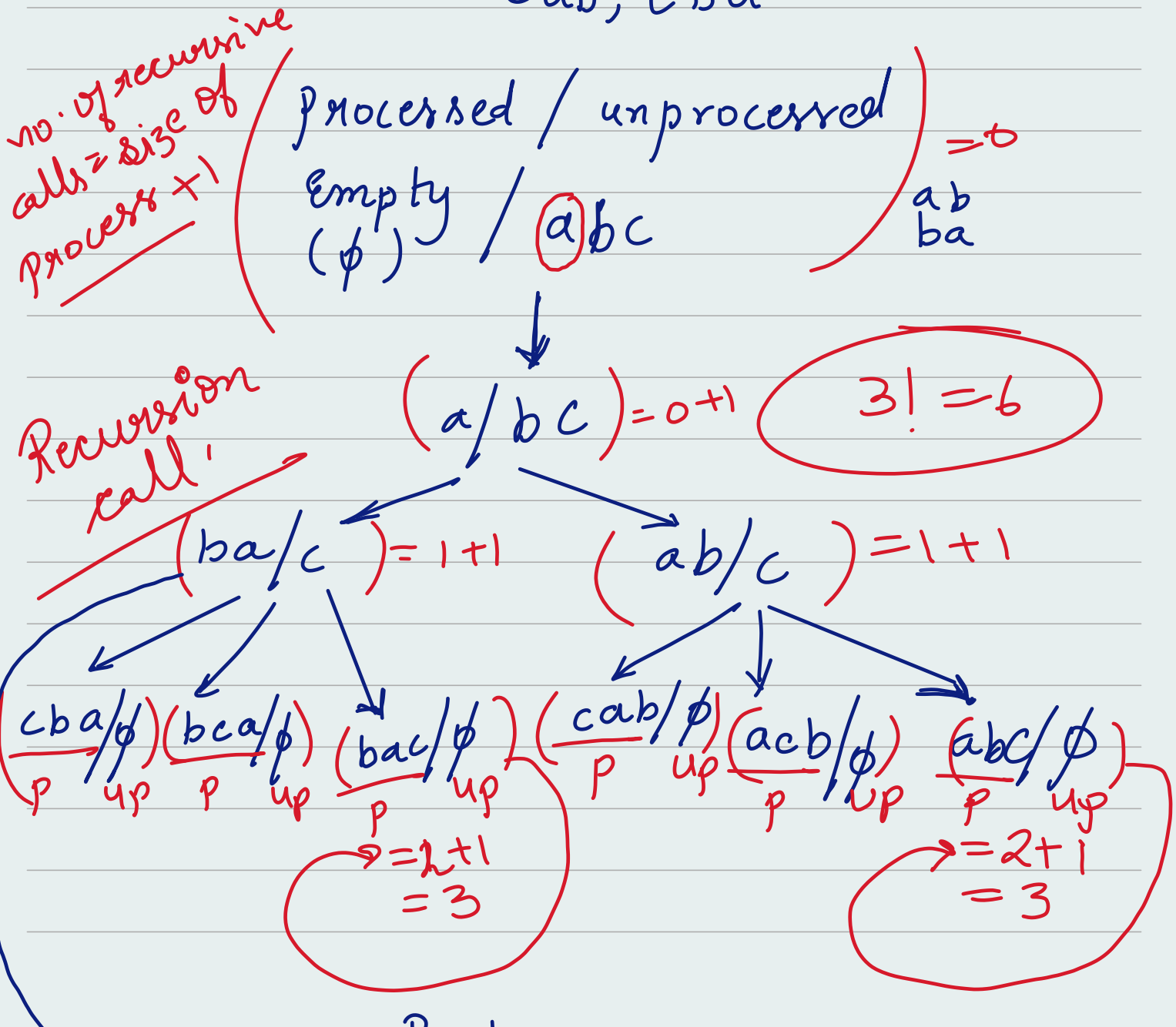
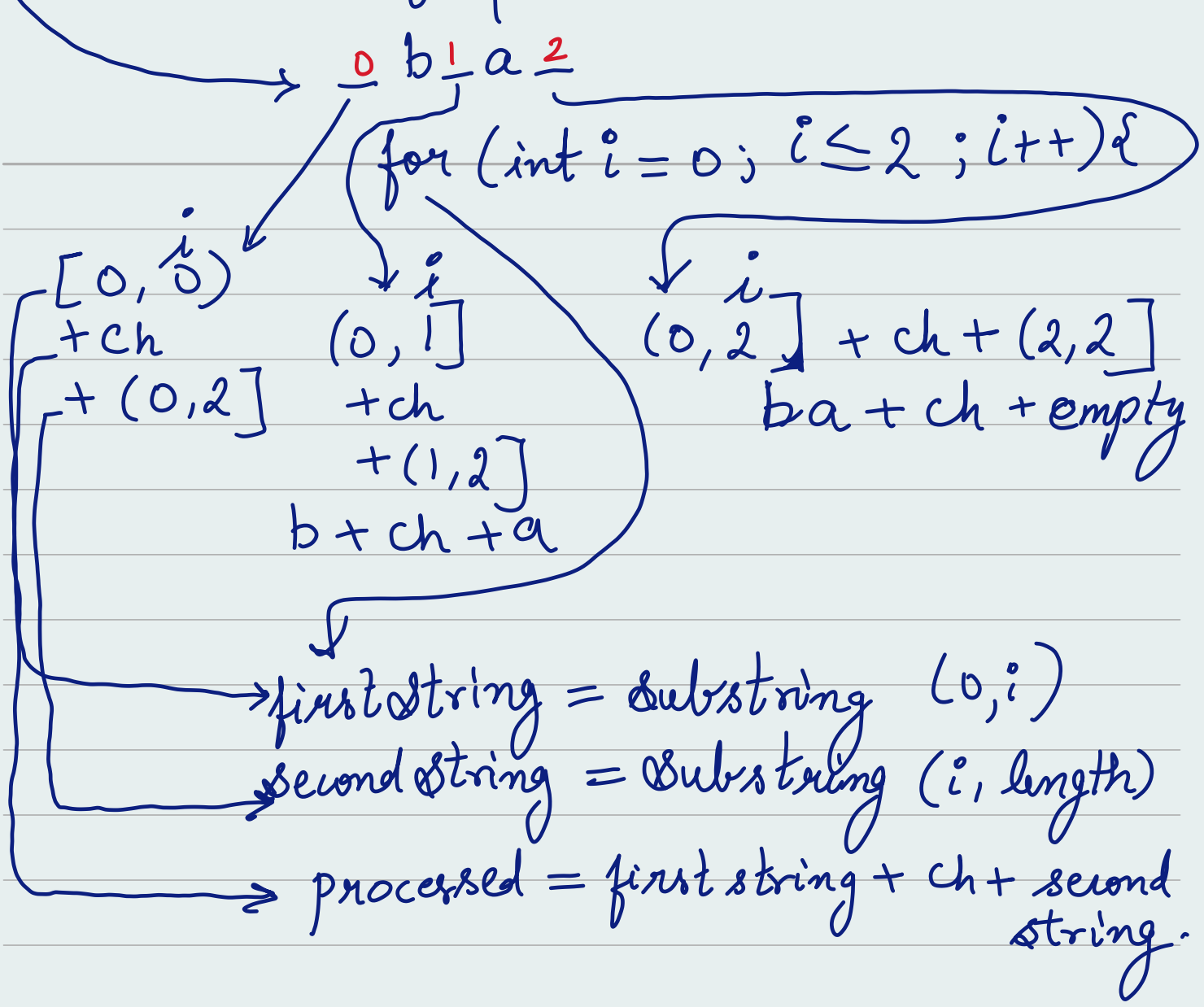
Recursion Permutations : Theory + Code

Str = a b c

permutation = { abc, acb, bac, bca, cab, cba }





Code:

```
psvm(){
    permutation("", "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 first = p.substring(0, i);
        string second = p.substring(i, p.length());
        permutation(first + ch + second,
                    up.substring(1));
    }
}
```

* Returning output as an array list & Counting Number of permutation

```
psvm() {
ArrayList<String> ans = permutationList("", "ABC");
System.out.println(ans);
}
static ArrayList<String> permutationList(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 first = p.substring(0, i);
        String second = p.substring(i, p.length());
        ans.addAll(permutationList(
```

```
        first + ch + second, up.substring
                (1));
    }
    return ans;
}
```

permutation count:

```
static int permutation_count(String p, String
                             up){
    if(up.isEmpty()){
        return 1;
    }
    char ch = up.charAt(0);
    int count = 0;

    for(int i=0; i<=p.length(); i++){
        String first = p.substring(0,i);
        String second = p.substring(i,p.length());
    count= count+permutationCount
            (first + ch + second, up
                .substring(1));
    return count
```

y^{y^y}