Recursion Permutations:
heory Odl
str=abc
poumutation = l'abc, acb, bac, bca, Cab, cba
no of recurring no of recurring Anocersed / unprocessed = 5 all raises (b) / abc ba
Recursion $(a/bc)=0+1$ $(3 =6)$
(ba/c)=1+1 $(ab/c)=1+1$
cba/d) (bca/d) (bac/d) (cab/p) (acb/d) (abc/d) - abc/d) - = 2+1
= 3

/ (for (int i = 0) $i \leq 2$; i++) s) (0,2] + ch + (2,2] ba + ch + empty +ch +(1,2 b+ch+a *first string = outstring (0,i)
second string = outstring (i, length) > processed = first string + ch + second string. ps vm () {
permutation (ps" ", up" "ABC"); static void permutations (string p, string up) & Chan $ch = up \cdot chandt(0);$ $for (int i = 0 \cdot i < = p \cdot length, i + +)$

string second = p & ubstring (0, i); stringsecond = p & ubstring (i, p. length ()); peumutation (first + ch + second up · Substring (;)); Returning output as an array list &

Country Number of permutation

Ps vm () &

Array list & string > ans = permutation list

(p: "", yp" Arec");

System out println (ans);

Staticproof permutation (string P, string up)

If (up. is Empty () &

Arraylist < string > list = new

Shruaylist <> C);

List. add (p);

greturn list;

first + ch + second, up substring 3 notium ans; permutation count. Static ent peum utation Count (string), string g) (up is Empty ()) {
return 1; chan ch = up chanAt (0); int count = 0; foul int 2=0; 2 <= p length (); 2++){ String forst = p. Substring (0,i); string second = P. Substring (Eplength) count= count+peumutationCount (first + ch + Second, up Substring (1)); return count



