**Permutations of a given string**

[recursion](http://www.practice.geeksforgeeks.org/tag-page.php?tag=recursion&isCmp=0)[string](http://www.practice.geeksforgeeks.org/tag-page.php?tag=string&isCmp=0)

[Amazon](http://www.practice.geeksforgeeks.org/tag-page.php?tag=Amazon&isCmp=1)

Given a string, print all permutations of a given string.

**Input:**

The first line of input contains an integer T denoting the number of test cases.  
Each test case contains a single string in capital letter.  
  
**Output:**

Print all permutations of a given string with single space and all permutations should be in lexicographically increasing order.  
  
**Constraints:**

1 ≤ T ≤ 10  
1 ≤ size of string ≤ 5  
  
**Example:**

**Input:**  
2  
ABC

ABSG

**Output:**  
ABC ACB BAC BCA CAB CBA

ABGS ABSG AGBS AGSB ASBG ASGB BAGS BASG BGAS BGSA BSAG BSGA GABS GASB GBAS GBSA GSAB GSBA SABG SAGB SBAG SBGA SGAB SGBA

\*\*For More Examples Use Expected Output\*\*

<http://www.practice.geeksforgeeks.org/problem-page.php?pid=382>

static boolean nextPermutation(char[] array) {

// Find longest non-increasing suffix

int i = array.length - 1;

while (i > 0 && array[i - 1] >= array[i])

i--;

// Now i is the head index of the suffix

// Are we at the last permutation already?

if (i <= 0)

return false;

// Let array[i - 1] be the pivot

// Find rightmost element that exceeds the pivot

int j = array.length - 1;

while (array[j] <= array[i - 1])

j--;

// Now the value array[j] will become the new pivot

// Assertion: j >= i

// Swap the pivot with j

char temp = array[i - 1];

array[i - 1] = array[j];

array[j] = temp;

// Reverse the suffix

j = array.length - 1;

while (i < j) {

temp = array[i];

array[i] = array[j];

array[j] = temp;

i++;

j--;

}

// Successfully computed the next permutation

return true;

}

public static void main(String[] args) {

// TODO code application logic here

Scanner sc = new Scanner(System.in);

int t = Integer.parseInt(sc.nextLine());

while(t-- > 0) {

String s = sc.nextLine();

char[] ch = s.toCharArray();

Arrays.sort(ch);

do{

String perm = new String(ch);

System.out.print(perm + " ");

}while(nextPermutation(ch));

System.out.println();

}

}