**Permutations of a given string :-**

Medium Accuracy: 34.65% Submissions: 204K+ Points: 4

Given a string **S**. The task is to print all **unique**permutations of the given string in lexicographically sorted order.

**Example 1:**

**Input:** ABC

**Output:**

ABC ACB BAC BCA CAB CBA

**Explanation:**

Given string ABC has permutations in 6

forms as ABC, ACB, BAC, BCA, CAB and CBA .

**Example 2:**

**Input:** ABSG

**Output:**

ABGS ABSG AGBS AGSB ASBG ASGB BAGS

BASG BGAS BGSA BSAG BSGA GABS GASB

GBAS GBSA GSAB GSBA SABG SAGB SBAG

SBGA SGAB SGBA

**Explanation:**

Given string ABSG has 24 permutations.

**Your Task:**  
You don't need to read input or print anything. Your task is to complete the function **find\_permutaion()**which takes the string S as input parameter and returns a vector of string in lexicographical order.

**Expected Time Complexity:**O(n! \* n)  
**Expected Space Complexity:**O(n! \* n)

**Constraints:**  
1 <= length of string <= 5

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution

{

public:

vector<string> find\_permutation(string s){

int n = s.size();

if(n==1) return {s};

int fact=1;

for(auto i=2; i<=n; i++)

fact \*= i;

set<string> st;

st.insert(s);

for(int i=2; i<=fact; i++){

next\_permutation(s.begin(), s.end());

st.insert(s);

}

int ind=0;

vector<string> ans(st.size());

for(auto str:st){

ans[ind]=str;

ind++;

}

return ans;

}

};

//{ Driver Code Starts.

int main(){

int t;

cin >> t;

while(t--)

{

string S;

cin >> S;

Solution ob;

vector<string> ans = ob.find\_permutation(S);

for(auto i: ans)

{

cout<<i<<" ";

}

cout<<"\n";

}

return 0;

}

// } Driver Code Ends

**T.C :- O(n!\*n)**

**S.C :- O(n!)**