**String Permutations :-**

Easy Accuracy: 48.33% Submissions: 24K+ Points: 2

Given a string S. The task is to find all permutations (**need not be different**) of a given string.

**Example 1:**

**Input:**

S = AAA

**Output:** AAA AAA AAA AAA AAA AAA  
**Explanation:** There are total 6 permutations, as given in the output.

**Example 2:**

**Input:**

S = 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:** There are total 24 permutations, as given in the output.

**Your Task:**  
This is a function problem. You only need to complete the **function permutation**that takes **S as parameter** and returns the list of **permutations**in lexicographically increasing order. The **newline is automatically**added by driver code.

**Constraints:**  
1 ≤ size of string ≤ 5

**Expected Time Complexity:** O(N \* N!), N = length of string.  
**Expected Auxiliary Space:** O(1)

**Code :-**

//{ Driver Code Starts

#include<bits/stdc++.h>

using namespace std;

// } Driver Code Ends

class Solution{

public:

void func(string &s, string temp, vector<string> &ans, int ind){

int n=s.size();

//base case

if(ind==n){

ans.push\_back(temp);

return;

}

for(auto i=0; i<n; i++){

if(s[i]!=0){

temp.push\_back(s[i]);

s[i] = 0;

func(s, temp, ans, ind+1);

s[i] = temp[ind];

temp.pop\_back();

}

}

return;

}

vector<string> permutation(string S){

vector<string> ans;

string temp="";

/\*

int fact=1;

for(auto i=S.size(); i>1; i--)

fact \*= i;

for(auto i=1; i<=fact; i++){

ans.push\_back(S);

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

}

\*/

func(S, temp, ans, 0);

sort(ans.begin(), ans.end());

return ans;

}

};

//{ Driver Code Starts.

int main()

{

int T;

cin>>T;

while(T--)

{

string S;

cin>>S;

Solution ob;

vector<string> vec = ob.permutation(S);

for(string s : vec){

cout<<s<<" ";

}

cout<<endl;

}

return 0;

}

// } Driver Code Ends

**T.C :- O(max{2^(N+1), N!})**

**S.C :- O(max{2^(N+1), N!})**