**Problem Name:** Abbreviations using backtracking **Tags:** Recursion, Backtracking, String, Math **Level:** Medium

**Language:** C++

**Problem Statement:** You are given a word, You have to generate all the abbreviations of that word using recursion as shown in below examples.

# Input Format:

The first and only line of each input contains a word.

# Output Format:

You have to print all the abbreviations as shown in sample output below.

# Constraints:

1 <= length of string <= 14

# Example:

Input:

abc

Output:

abc ab1 a1c a2 1bc 1b1 2c 3

# Solution:

**Explanation:** We can use another empty string at the beginning and iterate over each character in the input string and decide weather to include it in string or abbreviate it and increment the count. We can come to this conclusion from a few key points. One of the main key points is that the output contains 2length of input string Strings as answer. Meaning that we need to have an exponential solution that can decide on every step for each character in the input string.

# Code:

#include<iostream> using namespace std;

void solution(string str, string &asf,int count, int pos)

{

// write your code here if(pos == str.size()){

if(count == 0) cout<<asf<<endl;

else

cout<<asf<<count<<endl; return;

}

solution(str, asf + (count != 0 ? to\_string(count) : "" ) + str[pos], 0, pos + 1);

solution(str, asf, count + 1, pos + 1);

}

int main() { string str; cin >> str;

solution(str,"",0,0); return 0;

}

**Time Complexity:** O(2n)

**Space Complexity:** O(2n). We have taken an extra string which is modified in each iteration and for exponential iterations we will use exponential amount of extra space.