7)PRINT ALL PARTITIONS OF THE SUBSTRING:

LINK OF EXPLANATION: L17. Palindrome Partitioning | Leetcode | Recursion | C++ | Java

METHOD:

1) First of all, we have to check all the partitions in this question and we have no other choice.
2) And the main thing in this question is that if the substring before the partition is palindrome then only we go on partitioning the rest part of the string.
3) So in order to check whether the given substring is palindrome or not, we need to have the function.

   Palindrome function:

   ```
   bool isPalindrome(string &s,int start,int end){
       int i=start,j=end;
       while(i<=j){
           if(s[i]!=s[j]){
               return false;
           }
           i++;
           j--;
       }
       return true;
   }
   ```
4) The most important step is the backtracking step in the question which comes after the successful palindrome partition of the string.

   We pass the recursion for the rest of the string …..but after returning from there we have to pop the substring which was pushed into the string vector.

   The backtracking process:

   ```
   if(isPalindrome(s,start,i)==true){
     t.push_back(s.substr(start,i-start+1));
     function(s,i+1,answer,t);
     t.pop_back();
   }
   ```

CODE OF ABOVE METHOD:

```
#include<iostream>
#include<vector>
using namespace std;

bool isPalindrome(string &s,int start,int end){
   int i=start,j=end;
   while(i<=j){
      if(s[i]!=s[j]){
         return false;
      }
```

```
        i++;
        j--;
    }
    return true;
}

void function(string &s,int start,vector<vector<string>> &answer,vector<string> &t){
    if(start==s.length()){
        answer.push_back(t);
        return;
    }
    else{
        for(int i=start;i<s.length();i++){
            if(isPalindrome(s,start,i)==true){
                t.push_back(s.substr(start,i-start+1));
                function(s,i+1,answer,t);
                t.pop_back();
            }
        }
        return;
    }
}

vector<vector<string>> palindromePartitioning(string &s){
    vector<vector<string>> answer;
    vector<string> t;
    function(s,0,answer,t);
    return answer;
}

int main(){
    string s;
    cout<<"\n Enter the input string:";
    cin>>s;
    vector<vector<string>> answer=palindromePartitioning(s);
    cout<<"\n The palindrome partitioning of the string:";
    for(int i=0;i<answer.size();i++){
        cout<<"\n ";
        for(int j=0;j<answer[i].size();j++){
            cout<<answer[i][j]<<" ";
        }
    }
    return 0;
}
```