14) ALL PERMUTATIONS OF THE GIVEN STRING.

LINK OF EXPLANATION: Permutations of a String Explained with Code | String Permutatio.

METHOD:

1) In this question,we get to see that every character present in the input string takes all the positions at some or the other moment.
2) So the technique we use in this question is to place each and every character at every position of the string.
3) We first take all the characters of the input string in the unordered_map with their frequency.
4) The next thing we do is we iterate through the map and place each character at the particular index and while placing them, we decrement the frequency of that character by 1 and pass the recursion on the next index with the same map.
5) The backtracking step is that when the function returns the position on which we placed the character has to be replaced by another character and the frequency which was decreased earlier has to be increment by one
6) We do not call multiple times for the same character depending on the frequency.

CODE OF THE ABOVE PROGRAM:

```
class Solution
{
        public:
        void fun(string &t,int index,unordered_map<char,int> &hash,vector<string> &answer){
           if(index==t.size()){
              answer.push_back(t);
              return;
           }
           else{
              for(auto i=hash.begin();i!=hash.end();i++){
                 if(i->second!=0){
                    char temp=i->first;
                    i->second--;
                    t[index]=i->first;
                    fun(t,index+1,hash,answer);
                    hash[temp]++;
                 }
              }
              return;
           }
        }
   vector<string>find_permutation(string &s){
           unordered_map<char,int> hash;
           for(int i=0;i<s.length();i++){
              hash[s[i]]++;
           }
```

```
            string t=s;
            vector<string> answer;
            fun(t,0,hash,answer);
            sort(answer.begin(),answer.end());
            return answer;
    }
};
```