Permutations

An ordered arrangement of objects is called Permutations.

Permutation - 1

Given an array nums of **distinct** integers, print all the possible permutations.

Sample Test Cases:

Input: nums = [1,2,3]

Output: [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]

Input: nums = [0,1]

Output: [[0,1],[1,0]]

Idea:

You are at index idx, try out all the possible swap, Ensuring that you don't produce duplicates.

Code:

```
#include "bits/stdc++.h"
using namespace std;
void solve(vector<int> &nums, vector<vector<int>> &ans, int idx) {
    if (idx == nums.size()) {
        ans.push_back(nums);
        return ;
    }
    for (int i = idx; i < nums.size(); i++) {
        swap(nums[i], nums[idx]);
        solve(nums, ans, idx + 1);
        swap(nums[i], nums[idx]);
    }
}
vector<vector<int>> permute(vector<int> nums) {
    vector<vector<int>> ans;
    solve(nums,ans,0);
    return ans;
}
```

```
int32_t main() {
    vector<vector<int>> res = permute({1, 2, 3});
    for (auto i : res) {
        for (auto ii : i) {
            cout << ii << " ";
        }
        cout << "\n";
    }
}
```

Permutation - 2

Given a collection of numbers, nums, that might **contain duplicates**, print all possible unique permutations in any order.

Sample Test Case:
Input: nums = [1,1,2]
Output:
[[1,1,2], [1,2,1], [2,1,1]]

Idea:
While swapping, avoid duplicates. I.e avoid swapping the same numbers.

Code:

```cpp
#include "bits/stdc++.h"
using namespace std;
void solve(vector<int> nums, vector<vector<int>> &ans, int idx) {
    if (idx == nums.size()) {
        ans.push_back(nums);
        return ;
    }
    for (int i = idx; i < nums.size(); i++) {
        if(i != idx and nums[i]==nums[idx])
            continue;
        swap(nums[i], nums[idx]);
        solve(nums, ans, idx + 1);
    }
}
vector<vector<int>> permute(vector<int> nums) {
    vector<vector<int>> ans;
    sort(nums.begin(), nums.end());
    solve(nums,ans,0);
    return ans;
}
```

```
int32_t main() {
    vector<vector<int>> res = permute({1, 2, 2,2,3});
    for (auto i : res) {
        for (auto ii : i) {
            cout << ii << " ";
        }
        cout << "\n";
    }
}
```

Permutations using STL

Next_permutation(start,end):

If the function can determine the next higher permutation, it rearranges the elements as such and returns true. If that was not possible (because it is already at the largest possible permutation), it rearranges the elements according to the first permutation (sorted in ascending order) and returns false.

Code:

```cpp
#include "bits/stdc++.h"
using namespace std;
vector<vector<int>> permute(vector<int> nums) {
    vector<vector<int>> ans;
    sort(nums.begin(), nums.end());
    do {
        ans.push_back(nums);
    } while (next_permutation(nums.begin(), nums.end()));
    return ans;
}
int32_t main() {
    vector<vector<int>> res = permute({1, 2, 2});
    for (auto i : res) {
        for (auto ii : i) {
            cout << ii << " ";
        }
        cout << "\n";
    }
}
```