Our Solution(s) Run Code

Solution 1 Solution 2

```cpp
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

#include <vector>
using namespace std;

void permutationsHelper(vector<int> array, vector<int> currentP
                        vector<vector<int>> *permutations);

// Upper Bound: O(n^2*n!) time | O(n*n!) space
// Roughly: O(n*n!) time | O(n*n!) space
vector<vector<int>> getPermutations(vector<int> array) {
  vector<vector<int>> permutations;
  permutationsHelper(array, {}, &permutations);
  return permutations;
}

void permutationsHelper(vector<int> array, vector<int> currentP
                        vector<vector<int>> *permutations) {
  if (array.size() == 0 && currentPermutation.size() > 0) {
    permutations->push_back(currentPermutation);
  } else {
    for (int i = 0; i < array.size(); i++) {
      vector<int> newArray;
      newArray.insert(newArray.end(), array.begin(), array.begin
      newArray.insert(newArray.end(), array.begin() + i + 1, arr
      vector<int> newPermutation = currentPermutation;
      newPermutation.push_back(array[i]);
      permutationsHelper(newArray, newPermutation, permutations)
    }
  }
}

```

Your Solutions Run Code

Solution 1 Solution 2 Solution 3

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

vector<vector<int>> getPermutations(vector<int> array) {
  // Write your code here.
  return {};
}

```

Our Tests

Custom Output Submit Code

```
    for (vector<int> subArray : arrays) {
      if (array == subArray) {
        return true;
      }
    }
    return false;
}

class ProgramTest : public TestSuite {
public:
    void Run() {

      Run(test, "Test Case 1", [](){
        vector<vector<int>> parms = getParmsAsClass({});
        assert(parms.size() == 0);
      });

      Run(test, "Test Case 2", [](){
        vector<vector<int>> parms = getParmsAsClass({0});
        assert(parms.size() == 1);
        assert(isPermutation(parms, {0}));
```

Run or submit code when you're ready.