|  |  |
| --- | --- |
| **Partition in K subsets in C++** | |
| #include <iostream>  #include <vector>  using namespace std;  int counter = 0;  void solution(int i, int n, int k, int nos, vector<vector<int>>& ans) {      if (i > n) {          if (nos == k) {              counter++;              cout << counter << ". ";              for (auto& set : ans) {                  cout << "[";                  for (auto num : set) {                      cout << num << " ";                  }                  cout << "] ";              }              cout << endl;          }          return;      }      for (int j = 0; j < ans.size(); j++) {          if (!ans[j].empty()) {              ans[j].push\_back(i);              solution(i + 1, n, k, nos, ans);              ans[j].pop\_back();          } else {              ans[j].push\_back(i);              solution(i + 1, n, k, nos + 1, ans);              ans[j].pop\_back();              break;          }      }  }  int main() {      int n = 3;      int k = 2;      vector<vector<int>> ans(k);      solution(1, n, k, 0, ans);      return 0;  } | **Dry Run Table:**   | **Step** | **i** | **nos** | **ans (state)** | **Action Taken** | | --- | --- | --- | --- | --- | | 1 | 1 | 0 | [[], []] | Put 1 in first empty subset | | 2 | 2 | 1 | [[1], []] | Put 2 in subset 0 | | 3 | 3 | 1 | [[1, 2], []] | Put 3 in subset 0 | | 4 | 4 | 1 | — | nos != k, discard | | 5 | 3 | 2 | [[1, 2], [3]] | ✅ Output: [1 2] [3] | | 6 | 2 | 2 | [[1], [2]] | ✅ Output path starts | | 7 | 3 | 2 | [[1, 3], [2]] | ✅ Output: [1 3] [2] | | 8 | 3 | 2 | [[1], [2, 3]] | ✅ Output: [1] [2 3] |   **🧾 Final Output:**  1. [1 2 ] [3 ]  2. [1 3 ] [2 ]  3. [1 ] [2 3 ]  Top of Form  Bottom of Form |
| Output:- 1. [1 2 ] [3 ]  2. [1 3 ] [2 ]  3. [1 ] [2 3 ] | |