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 = 4;
   int k = 3;
   vector<vector<int>> ans(k);

   solution(1, n, k, 0, ans);

   return 0;
}
```

Step-by-step Execution:

1. i = 1:
   - Try placing 1 in the first subset:
     - Add 1 to ans[0] → ans = [[1], [], []].
     - Recursively call solution(2, 4, 3, 1, ans).

2. i = 2:
   - Try placing 2 in the first subset:
     - Add 2 to ans[0] → ans = [[1, 2], [], []].
     - Recursively call solution(3, 4, 3, 1, ans).
   - Try placing 2 in the second subset:
     - Add 2 to ans[1] → ans = [[1], [2], []].
     - Recursively call solution(3, 4, 3, 2, ans).

3. i = 3:
   - For the current state of ans:
     - For ans[0] = [1, 2]:
       - Try placing 3 in the first subset → ans = [[1, 2, 3], [], []].
       - Recursively call solution(4, 4, 3, 1, ans).
     - For ans[1] = [2]:
       - Try placing 3 in ans[1] → ans = [[1], [2, 3], []].
       - Recursively call solution(4, 4, 3, 2, ans).

4. i = 4:
   - Now, the subsets are filled with i = 1, 2, 3 elements.
   - After backtracking, we update the subsets and print the results when nos == k.

Final Outputs (Valid Partitions):

1. First partition:
   - ans = [[1, 2], [3], [4]]
   - Output: 1. [1 2] [3] [4]
2. Second partition:

<table>
<tr>
<td></td>
<td>

ans = [[1, 3], [2], [4]]
Output: 2. [1 3] [2] [4]

3. Third partition:

ans = [[1], [2, 3], [4]]
Output: 3. [1] [2 3] [4]

4. Fourth partition:

ans = [[1, 4], [2], [3]]
Output: 4. [1 4] [2] [3]

5. Fifth partition:

ans = [[1], [2, 4], [3]]
Output: 5. [1] [2 4] [3]

6. Sixth partition:

ans = [[1], [2], [3, 4]]
Output: 6. [1] [2] [3 4]

</td>
</tr>
<tr>
<td colspan="2">Output:-
1. [1 2] [3] [4]
2. [1 3] [2] [4]
3. [1] [2 3] [4]
4. [1 4] [2] [3]
5. [1] [2 4] [3]
6. [1] [2] [3 4]</td>
</tr>
</table>