Gray Code in C++

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

void backtrack(vector<int>& ans, int n, int& temp) {
    if (n == 0) {
        ans.push_back(temp);
        return;
    }

    backtrack(ans, n - 1, temp);

    temp = temp ^ (1 << (n - 1));
    backtrack(ans, n - 1, temp);
}

vector<int> grayCode(int n) {
    vector<int> ans;
    if (n == 0) {
        ans.push_back(0);
        return ans;
    }

    int temp = 0;
    backtrack(ans, n, temp);
    return ans;
}

int main() {
    vector<int> ans = grayCode(3);
    sort(ans.begin(), ans.end());

    for (int num : ans) {
        cout << num << " ";
    }
    cout << endl;

    return 0;
}
```

Gray Code Summary

- A Gray code of n bits is a sequence of 2^n integers where **each successive number differs by only one bit**.
- This implementation generates it recursively by flipping one bit at each step using XOR:
  temp = temp ^ (1 << (n - 1))

Dry Run: grayCode(3)

We'll track:

Call Depth	n	temp (Decimal)	temp (Binary)	Action
0	3	0	000	call (3→2)
1	2	0	000	call (2→1)
2	1	0	000	call (1→0)
3	0	0	000	push 0
2	1	1	001	flip bit 0 → 1
3	0	1	001	push 1
1	2	3	011	flip bit 1 → 1
2	1	3	011	call (1→0)
3	0	3	011	push 3
2	1	2	010	flip bit 0 → 0
3	0	2	010	push 2
0	3	6	110	flip bit 2 → 1
1	2	6	110	call (2→1)
2	1	6	110	call (1→0)
3	0	6	110	push 6
2	1	7	111	flip bit 0 → 1
3	0	7	111	push 7
1	2	5	101	flip bit 1 → 0
2	1	5	101	call (1→0)

3	0	5	101	push 5
2	1	4	100	flip bit 0 → 0
3	0	4	100	push 4

✅ **Generated Sequence (before sort):**

{0, 1, 3, 2, 6, 7, 5, 4}

0 1 2 3 4 5 6 7