Josephus in C++

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

int solution(int n, int k) {
   if (n == 1) {
      return 0;
   }
   int x = solution(n - 1, k);
   int y = (x + k) % n;
   return y;
}

int main() {
   int n = 4;
   int k = 2;
   cout << solution(n, k) << endl;
   return 0;
}
```

Step-by-Step Execution:

The function uses recursion to solve the Josephus problem. The base case is when n = 1, where the last remaining person is at position 0. The recursive case computes the position of the last person standing for n-1 people and then adjusts it by the step k using modulo operation.

1. **Initial Call**: solution(4, 2)
   - **n = 4**, **k = 2**
   - Call solution(3, 2) (since n - 1 = 3)

2. **Second Call**: solution(3, 2)
   - **n = 3**, **k = 2**
   - Call solution(2, 2) (since n - 1 = 2)

3. **Third Call**: solution(2, 2)
   - **n = 2**, **k = 2**
   - Call solution(1, 2) (since n - 1 = 1)

4. **Base Case**: solution(1, 2)
   - **n = 1**, **k = 2** (base case)
   - Return 0 (last remaining person at position 0)

5. **Returning from Third Call**: solution(2, 2)
   - Result from solution(1, 2) is 0
   - Calculate y = (0 + 2) % 2 = 0
   - Return y = 0

6. **Returning from Second Call**: solution(3, 2)
   - Result from solution(2, 2) is 0
   - Calculate y = (0 + 2) % 3 = 2
   - Return y = 2

7. **Returning from First Call**: solution(4, 2)
   - Result from solution(3, 2) is 2
   - Calculate y = (2 + 2) % 4 = 0
   - Return y = 0

<table>
<tr><td></td><td><b>Final Output:</b>
The position of the last remaining person (zero-indexed) is 0, so the output is 0.</td></tr>
<tr><td colspan="2">Output:-
0</td></tr>
</table>