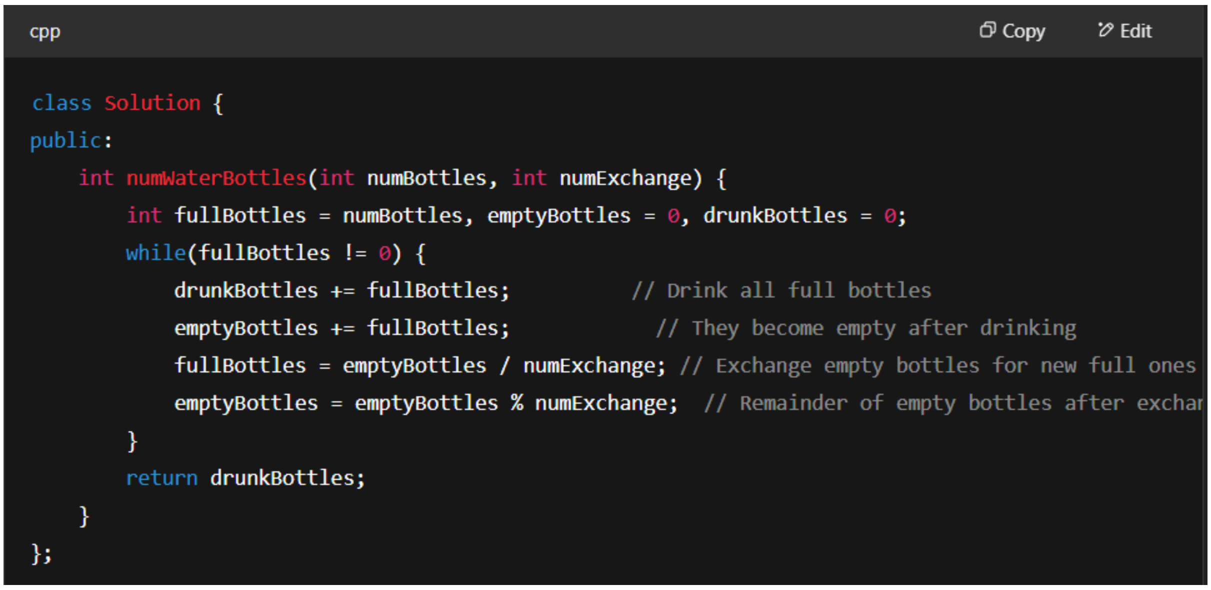
Reasoned about water bottle exchange logic for a couple of seconds

The provided solution correctly implements the iterative process to calculate the maximum number of water bottles that can be drunk. Here's a brief explanation of the code's logic:

- **Initialization:**
  - fullBottles starts as the initial number of full bottles (i.e. numBottles).
  - emptyBottles is initially 0.
  - drunkBottles (the count of bottles drunk) is also initially 0.
- **While Loop:**
  - The loop continues as long as there is at least one full bottle available.
  - On each iteration, all full bottles are drunk (added to drunkBottles), and each consumed bottle becomes empty (added to emptyBottles).
  - Next, the available empty bottles are exchanged for new full bottles using integer division with numExchange.
  - Finally, the remaining empty bottles (those that weren't enough for an exchange) are updated using the modulus operation.
- **Termination:**
  - When there are no more full bottles available to drink (i.e. fullBottles becomes 0), the loop terminates and the total drunkBottles is returned.

Below is the complete code for clarity:



```cpp
class Solution {
public:
    int numWaterBottles(int numBottles, int numExchange) {
        int fullBottles = numBottles, emptyBottles = 0, drunkBottles = 0;
        while(fullBottles != 0) {
            drunkBottles += fullBottles;          // Drink all full bottles
            emptyBottles += fullBottles;            // They become empty after drinking
            fullBottles = emptyBottles / numExchange; // Exchange empty bottles for new full ones
            emptyBottles = emptyBottles % numExchange;  // Remainder of empty bottles after exchar
        }
        return drunkBottles;
    }
};
```

This implementation efficiently handles the problem and works for various inputs.