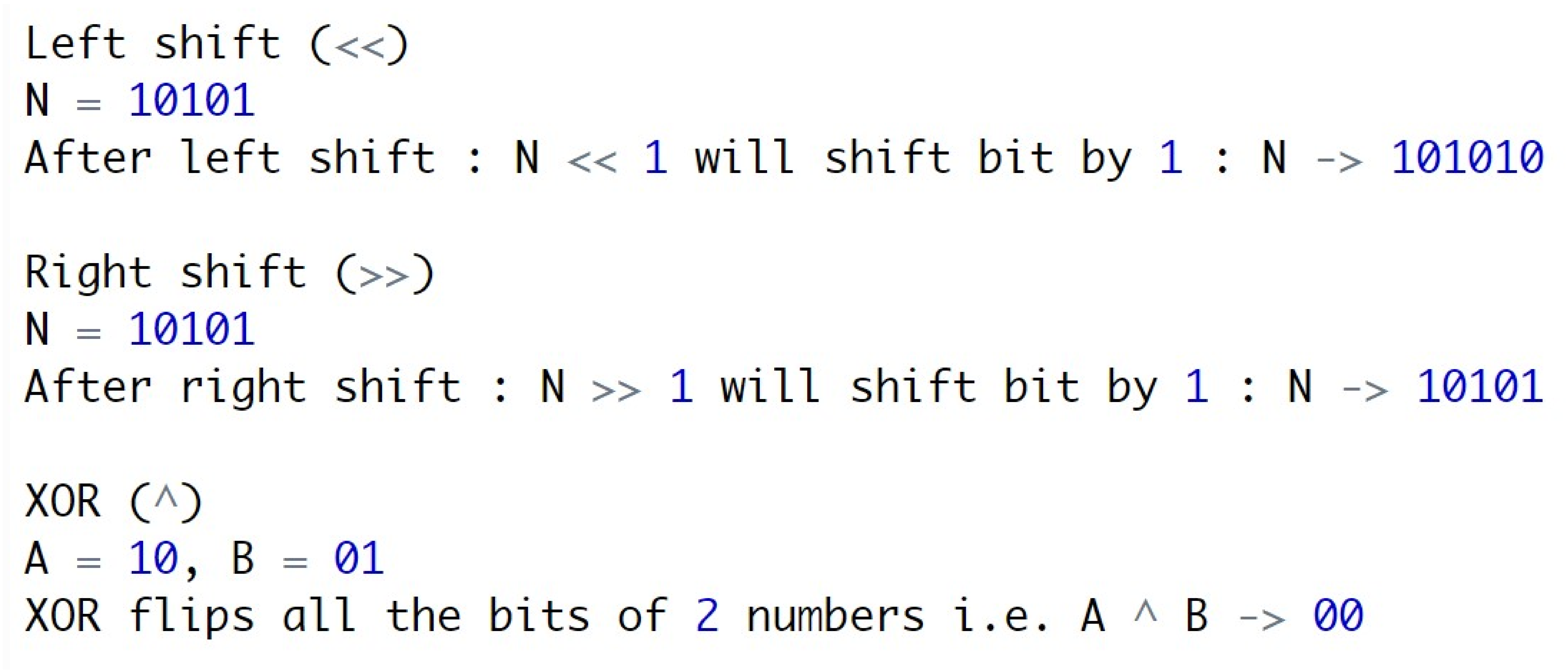
Bit manipulation important concepts!

~ Prakhar Rai

#1 Some basic steps

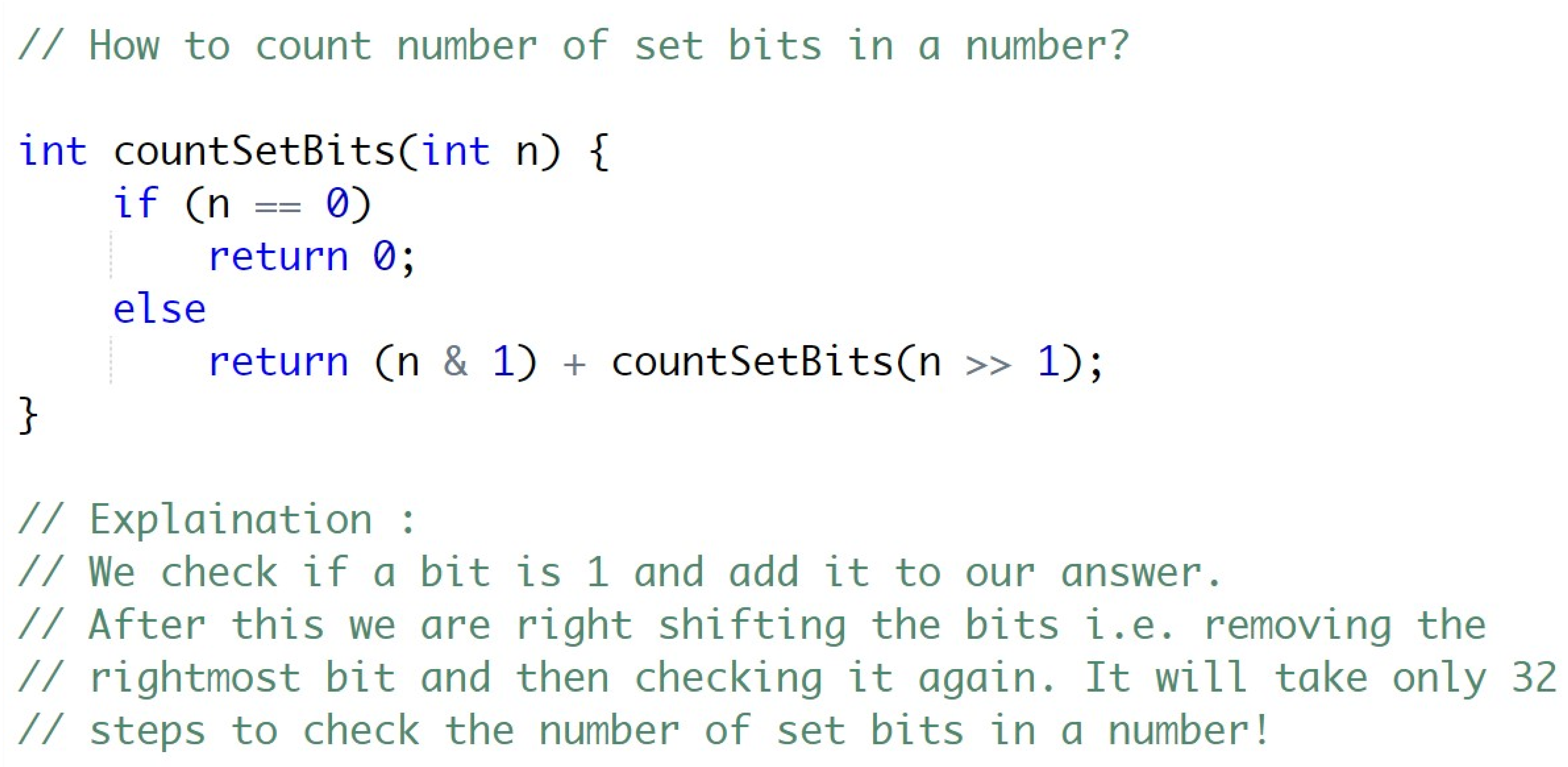


```
Left shift (<<)
N = 10101
After left shift : N << 1 will shift bit by 1 : N -> 101010

Right shift (>>)
N = 10101
After right shift : N >> 1 will shift bit by 1 : N -> 10101

XOR (^)
A = 10, B = 01
XOR flips all the bits of 2 numbers i.e. A ^ B -> 00
```

#2 Count the number of set bits in a number.

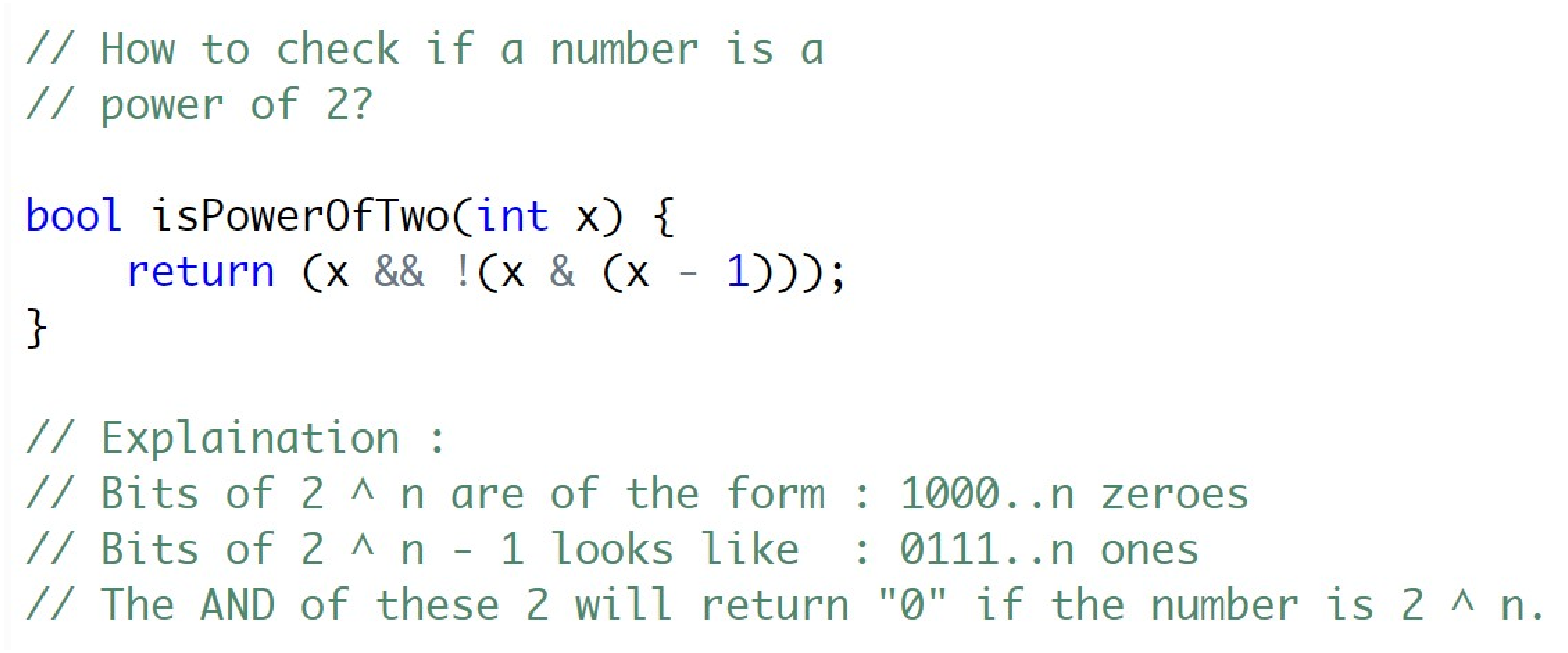


```
// How to count number of set bits in a number?

int countSetBits(int n) {
    if (n == 0)
        return 0;
    else
        return (n & 1) + countSetBits(n >> 1);
}

// Explaination :
// We check if a bit is 1 and add it to our answer.
// After this we are right shifting the bits i.e. removing the
// rightmost bit and then checking it again. It will take only 32
// steps to check the number of set bits in a number!
```

#3 How to check if a number is a power of 2?

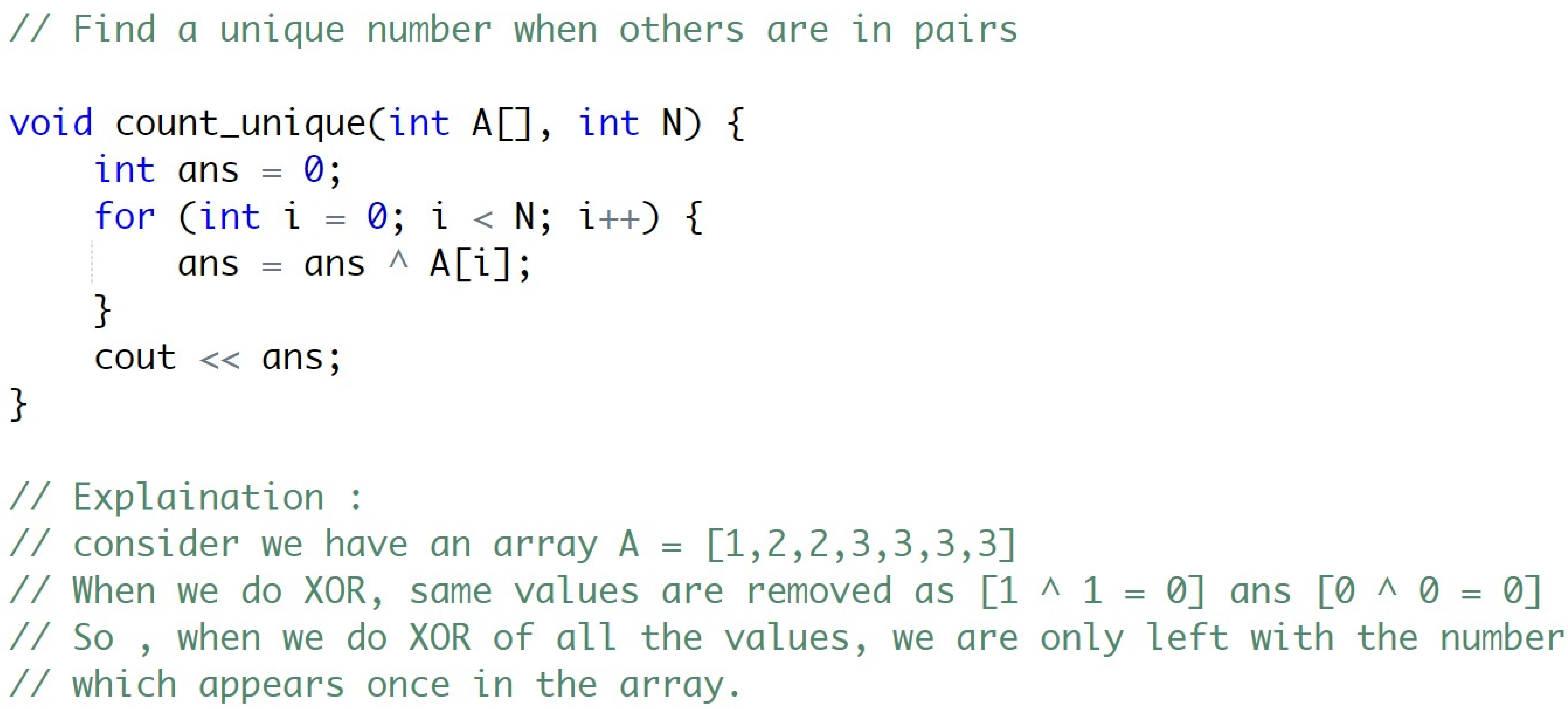


```
// How to check if a number is a
// power of 2?

bool isPowerOfTwo(int x) {
    return (x && !(x & (x - 1)));
}

// Explaination :
// Bits of 2 ^ n are of the form : 1000..n zeroes
// Bits of 2 ^ n - 1 looks like  : 0111..n ones
// The AND of these 2 will return "0" if the number is 2 ^ n.
```

#4 Find a unique number when others are in pairs.

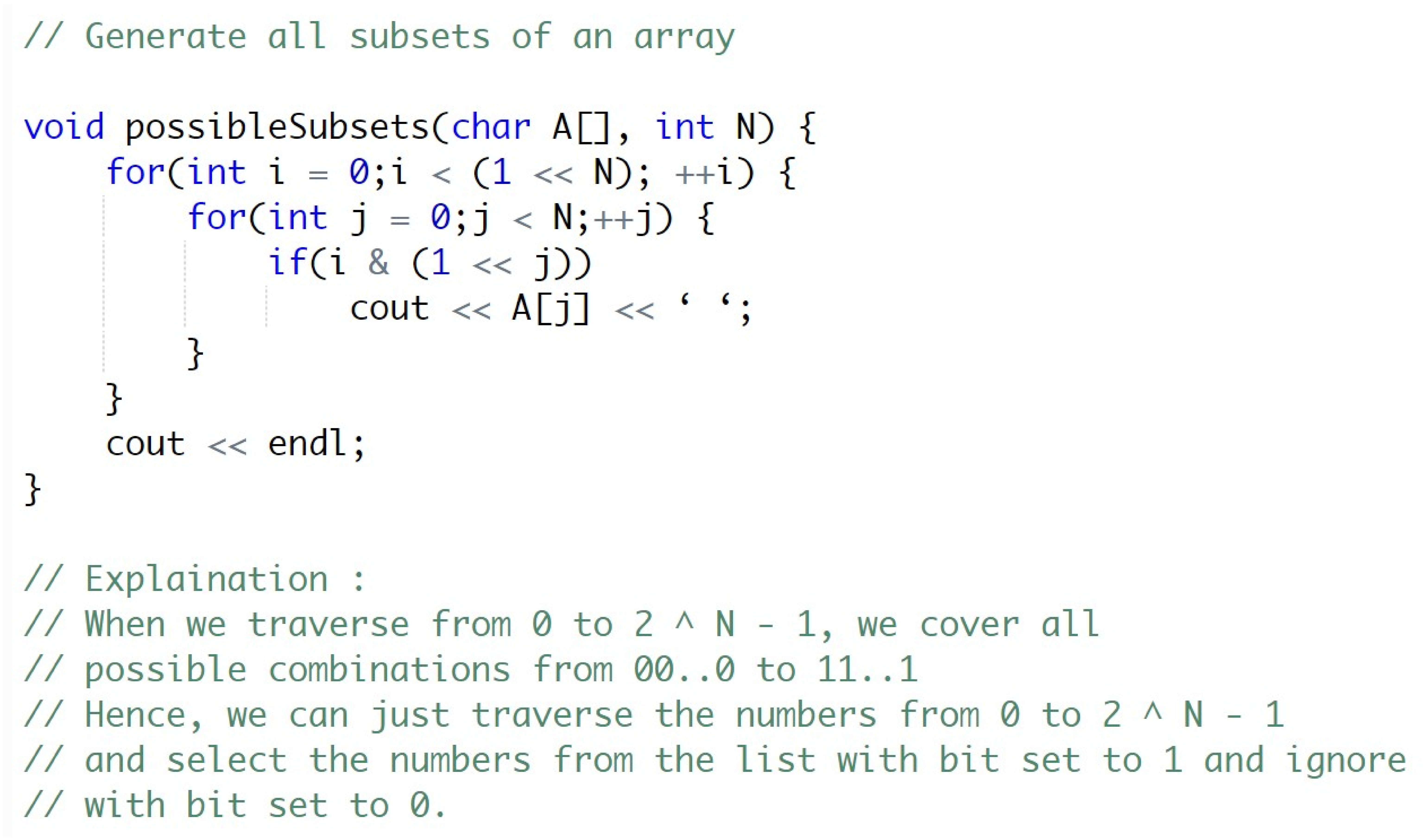


```
// Find a unique number when others are in pairs

void count_unique(int A[], int N) {
    int ans = 0;
    for (int i = 0; i < N; i++) {
        ans = ans ^ A[i];
    }
    cout << ans;
}

// Explaination :
// consider we have an array A = [1,2,2,3,3,3,3]
// When we do XOR, same values are removed as [1 ^ 1 = 0] ans [0 ^ 0 = 0]
// So , when we do XOR of all the values, we are only left with the number
// which appears once in the array.
```

#5 Generate all subsets

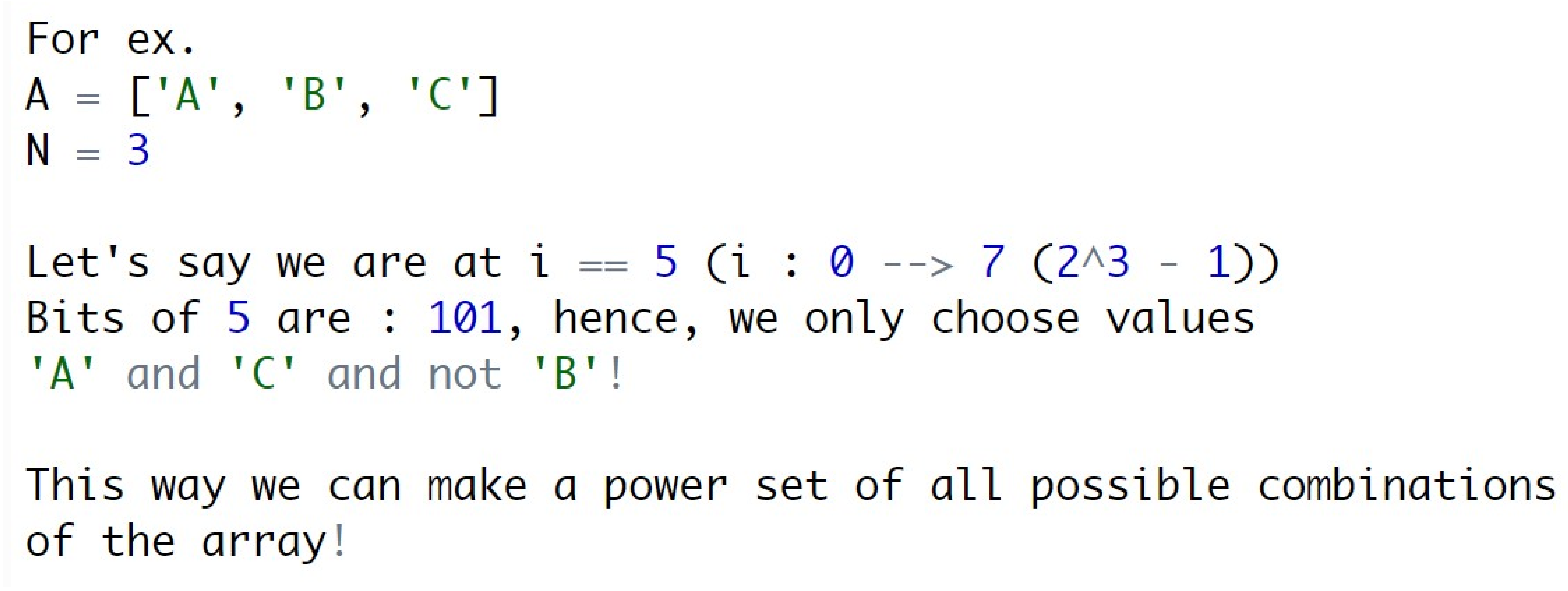


```
// Generate all subsets of an array

void possibleSubsets(char A[], int N) {
    for(int i = 0;i < (1 << N); ++i) {
        for(int j = 0;j < N;++j) {
            if(i & (1 << j))
                cout << A[j] << ‘ ‘;
        }
    }
    cout << endl;
}

// Explaination :
// When we traverse from 0 to 2 ^ N - 1, we cover all
// possible combinations from 00..0 to 11..1
// Hence, we can just traverse the numbers from 0 to 2 ^ N - 1
// and select the numbers from the list with bit set to 1 and ignore
// with bit set to 0.
```

#5 Generate all subsets (contd)



```
For ex.
A = ['A', 'B', 'C']
N = 3

Let's say we are at i == 5 (i : 0 --> 7 (2^3 - 1))
Bits of 5 are : 101, hence, we only choose values
'A' and 'C' and not 'B'!

This way we can make a power set of all possible combinations
of the array!
```