```cpp
#include <bits/stdc++.h>
using namespace std;
#define ll long long
```

Useful bits defines:

```cpp
#define log(x , y) log10(y)/log10(x)
#define log2(x) (31^__builtin_clz(x))
#define log2(x) (63^__builtin_clzll(x))
#define bit_count(x) __builtin_popcount(x);
```

Brute Force Bitmask:

```cpp
// to choose and check bitmask:
// for(int mask=0 ; mask<1<<n ; mask++) {
//     for(int i=0 ; i<n ; i++) {
//         if(mask&1<<i) {
//          // ...
//         }
//     }
// }
```

Convert binary to decimal:

```cpp
ll toDecimal(string binary){
    unsigned long long decimal = std::bitset<31>(binary).to_ulong(); //change the number of bits
    return decimal;
}

// or
ll binaryToDecimal(vector <ll> a){
    ll sum = 0;
    for (ll i = 0; i < a.size(); i++)
        sum+= (a[i] * 1LL<<i);
    return sum;
}
```

Convert decimal to binary:

```cpp
string toBinary(int n){
    std::string binary = std::bitset<31>(n).to_string(); //change number of bits
    for (int i = 0; i < binary.size(); i++)
        if(binary[i]=='1'){
            binary.erase(binary.begin() , binary.begin()+i);
            break;
        }
    return binary;
}
```

Xnor bitwise operation:

```cpp
int xnor(int x, int y){
    if (x < y) swap(x, y);
    if (x == 0 && y == 0) return 1;
    int a_rem = 0 , b_rem = 0, count = 0 , xnornum = 0;
    while (x){
        a_rem = x & 1; b_rem = y & 1;
        if (a_rem == b_rem) xnornum |= (1 << count);
        count++;
        x = x >> 1; y = y >> 1;
    }
    return xnornum;
}
```