bitmask DP

```
for (int mask = 0; mask < (1 << n); mask++) {
    for (int submask = mask; submask != 0; submask = (submask - 1) & mask) {
        int subset = mask ^ submask;
        // do whatever you need to do here
    }
}
The goal of this problem is to partition the elements into sets such that (any condition we will dp on it).
```

```
#define ON(n, k)        ((n) | (1 << (k)))         // Set k-th bit ON
#define OFF(n, k)       ((n) & ~(1 << (k)))        // Set k-th bit OFF
#define isON(n, k)      (((n) >> (k)) & 1)         // Check if k-th bit is ON
#define isPow2(n)       (!(n & (n - 1)))           // Check if number is power of two

int addJthBit(int x, int bit) {
    return x | (1 << bit);                         // Turn bit ON
}
int removeJthBit(int x, int bit) {
    return x & ~(1 << bit);                        // Turn bit OFF
}
bool checkBit(int x, int bit) {
    return (x >> bit) & 1;                         // Return true if bit is set
}
int toggleBit(int x, int bit) {
    return x ^ (1 << bit);                         // Toggle bit
}
int allBitsOne(int numberOfBits) {
    return (1 << numberOfBits) - 1;                // Return mask of all 1s
}
bool checkPowerOfTwo(int x) {
    return !(x & (x - 1));                         // Another way to check power of 2
}
bool isDivisibleByPowerOf2(int n, int k) {
    int powerOf2 = 1 << k;
    return (n & (powerOf2 - 1)) == 0;
}
string toBinary(int x) {
    string res;
    while (x) {
        res += char((x % 2) + '0');
        x >>= 1;
    }
    while (res.size() < 4) res += '0';             // Pad to 4 bits if needed
    reverse(res.begin(), res.end());
    return res;
}
void go() {
    int n, x;
    cin >> n >> x;
    int a[n + 2];
    for (int i = 0; i < n; i++) cin >> a[i];

    int lim = (1 << n);
    for (int msk = 0; msk < lim; msk++) {
        int sum = 0;
        cout << "Current Mask = " << msk << '\n';
        cout << toBinary(msk) << '\n';
```

```
        for (int elm = 0; elm < n; elm++) {
            cout << "Current element index = " << elm
                 << " status = " << checkBit(msk, elm) << '\n';
            if (checkBit(msk, elm))
                sum += a[elm];
        }
        cout << "Sum of the current Mask = " << sum << '\n';
        if (sum == x) {
            cout << "Found\nMask = " << msk << '\n';
            return;
        }
    }
    cout << "Not Found\n";
}
```

`bitset` Cheat Sheet in C++

```
#include <bitset>

// Initialization Examples
bitset<8> b;               // 8 bits, all initialized to 0
bitset<8> b("10101010");   // from binary string (rightmost is LSB)
bitset<8> b(42);           // from integer value
```

Basic Operations

Operation	Description	Example
`b.set()`	Set all bits to 1	`b.set();`
`b.set(i)`	Set bit at position `i` to 1	`b.set(2);`
`b.set(i, v)`	Set bit `i` to value `v` (0 or 1)	`b.set(2, 0);`
`b.reset()`	Set all bits to 0	`b.reset();`
`b.reset(i)`	Set bit at position `i` to 0	`b.reset(2);`
`b.flip()`	Flip all bits	`b.flip();`
`b.flip(i)`	Flip bit at position `i`	`b.flip(2);`

Query & Utility Operations

Operation	Description	Example
`b.test(i)`	Check if bit `i` is set (`bool`)	`b.test(3)`
`b[i]`	Access bit at position `i` (read/write)	`b[2] = 1;`
`b.count()`	Count of bits set to 1	`b.count();`
`b.size()`	Total number of bits	`b.size();`
`b.any()`	True if any bit is 1	`b.any();`
`b.none()`	True if all bits are 0	`b.none();`
`b.all()`	True if all bits are 1	`b.all();`
`b.to_ulong()`	Convert to `unsigned long`	`b.to_ulong();`
`b.to_ullong()`	Convert to `unsigned long long`	`b.to_ullong();`
`b.to_string()`	Convert to string of 0s and 1s	`b.to_string();`