SUBSUMS - Subset Sums

[#binary-search](https://www.spoj.com/problems/tag/binary-search) [#bitmasks](https://www.spoj.com/problems/tag/bitmasks)

Given a sequence of N (1 ≤ N ≤ 34) numbers S1, ..., SN (-20,000,000 ≤ Si ≤ 20,000,000), determine how many subsets of S (including the empty one) have a sum between A and B (-500,000,000 ≤ A ≤ B ≤ 500,000,000), inclusive.

Input

The first line of standard input contains the three integers N, A, and B. The following N lines contain S1 through SN, in order.

Output

Print a single integer to standard output representing the number of subsets satisfying the above property. Note that the answer may overflow a 32-bit integer.

Example

**Input:**

3 -1 2

1

-2

3

**Output:**

5

The following 5 subsets have a sum between -1 and 2:

* 0 = 0 (the empty subset)
* 1 = 1
* 1 + (-2) = -1
* -2 + 3 = 1
* 1 + (-2) + 3 = 2

#include<bits/stdc++.h>

#define ll int64\_t

#define maxm 1e6

#define f(i, in, n) for(int64\_t i=in; i<n; i++)

#define fl(i, in, n) for(int64\_t i=n-1; i>=in; i--)

#define lim(i, a, b) for (int64\_t i=a; i<b; i++)

using namespace std;

void no\_of\_subsets(vector<ll> v, ll i, ll sum, vector<ll> &x) {

    if (i>=v.size()) {

        x.push\_back(sum);

        return;

    }

    no\_of\_subsets(v, i+1, sum+v[i], x);

    no\_of\_subsets(v, i+1, sum, x);

}

int main() {

    ios\_base::sync\_with\_stdio(false);

    cin.tie(NULL);

    /\*int q;

    cin>>q;

    while (q--) {

    }\*/

    ll n, a, b;

    cin>>n>>a>>b;

    vector<ll> v(n);

    f(i, 0, n) cin>>v[i];

    vector<ll> x, y, v1, v2;

    f(i, 0, n/2) v1.push\_back(v[i]);

    lim(i, n/2, n) v2.push\_back(v[i]);

    no\_of\_subsets(v1, 0, 0, x);

    no\_of\_subsets(v2, 0, 0, y);

    sort(x.begin(), x.end());

    sort(y.begin(), y.end());

    ll ct=0;

    f(i, 0, x.size()) {

        ll l=lower\_bound(y.begin(), y.end(), a-x[i])-y.begin();

        ll r=upper\_bound(y.begin(), y.end(), b-x[i])-y.begin();

        ct+=(r-l);

    }

    cout<<ct;

    return 0;

}