# C1-B problem

## 题目描述

Zhoues 对《算法导论》这本书简直是爱不释手，当他翻到这本书的第 23 页准备暗中观察时，立马被 Horner 规则给吸引到了，于是他打算做一个基于 Horner 规则的一元多项式计算器，帮助他进行一类特殊的二元多项式。

具体来说，给定如下两个一元多项式：

$$\sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

$$\sum_{i=0}^{m} b_i y^i = b_0 + b_1 y + b_2 y^2 + \ldots + b_m y^m$$

定义如下的二元多项式：

$$f(x,y) = \left(\sum_{i=0}^{n} a_i x^i\right)\left(\sum_{i=0}^{m} b_i y^i\right)$$

你需要处理 q 次计算，第 i 次计算给定两个变量值 XiXi 和 YiYi，你需要求解 f(Xi,Yi)mod10007。

## 输入格式及数据规模

第一行一个正整数 n（1≤n≤3×10^4），表示第一个一元多项式的次数。

第二行 n+1 个非负整数 a0,a1,...,ana0,a1,...,an（0≤ai≤10^3），表示第一个一元多项式的系数。

第三行一个正整数 m（1≤m≤3×10^4），表示第二个一元多项式的次数。

第四行 m+1 个非负整数 b0,b1,...,bnb0,b1,...,bn（0≤bi≤10^3），表示第二个一元多项式的系数。

第五行一个正整数 q（1≤q≤2.5×10^3），表示计算的次数。

接下来 q行，第 i 行两个非负整数 Xi,Yi（0≤Xi,Yi≤10^4），表示第 i 次计算给定的两个变量值。

## 题解思路

- 为了避免计算过程中数值过大越界，利用提示：$ab \bmod c = (a \bmod c)(b \bmod c) \bmod c$，在每次计算两数乘法时先将两数分别取模相乘后再取模。
- 由提示，可以分别计算两个一元多项式对10007取模后相乘

- 对于每个一元多项式，计算每项模值之和，由于次数最大可取到3*10^4，若每项相乘再相加计算，算法的时间复杂度将为O(n^2)，实测会超时。
- 对算法进行优化：发现一元多项式次数一次递增，因此一元多项式每项的x^i可以使用前一项的模值计算。可以将算法复杂度降为O(n)。

# 代码

```
#include <iostream>
#include <math.h>
#include <vector>

using namespace std;

const int MOD = pow(10,4) + 7;

int main() {
    int n1, n2;
    vector<int> v1, v2;
    cin >> n1;
    for (int i = 0; i <= n1; i++) {
        int tmp;
        cin >> tmp;
        v1.push_back(tmp);
    }
    cin >> n2;
    for (int i = 0; i <= n2; i++) {
        int tmp;
        cin >> tmp;
        v2.push_back(tmp);
    }
    int n;
    cin >> n;
    for (int i = 0; i < n; i++) {
        int x, y;
        cin >> x;
        cin >> y;

        int ans_x = v1[0], ans_y = v2[0];
        int ans = 0;
        int x_p = 1, y_p = 1;
        for (int j = 1; j < v1.size(); j++) {
            x_p = (x_p * x) % MOD;
            ans_x += ((v1[j] * x_p) % MOD);
            ans_x %= MOD;
        }
        for (int j = 1; j < v2.size(); j++) {
            y_p = (y_p * y) % MOD;
            ans_y += ((v2[j] * y_p) % MOD);
            ans_y %= MOD;
        }
        ans = (ans_y * ans_x) % MOD;
        cout << ans << endl;
    }
    return 0;
```

```
}
```