CITRC 程式設計組期末競賽 第3屆 題解

A. 崩壞的代理商與無法取得的特典 Unattainable

> **難度**
> *Hard*

> **考點**
> 排序與二分搜

轉換一下問題

要拿到僅有 x 份的特點 $\rightarrow$ 比第 x 個人還要早到

- 對於 A 影院
  最大的（最晚的）m $\rightarrow$ 第 x 個人到的時間少一就是答案
- 對於 B 影院
  最大的（最晚的）m $\rightarrow$ 最大的 m 使得 $f(m) < x$

Subtask 2. 去 A 影城且 $< t_n >$ 由小到大排序

> 目標：比第 x 個人還要早到，第 x 個人到的時間少一就是答案

第 x 個人到的時間是 $t[x]$，所以 $t[x] - 1$ 就是答案

```
for(int i=1;i<=n;i++) cin >> t[i];
cout << t[x]-1;
```

Subtask 5. 去 A 影城且 $< t_n >$ 不會照大小排

不會照大小排怎麼辦？肯定是讓他照大小排
怎麼做呢？用上課教過的 `sort`

```
for(int i=1;i<=n;i++) cin >> t[i];
sort(t+1,t+n+1);
cout << t[x]-1;
```

Subtask 3. 去 B 影城且 $a = b = 0$

$a = b = 0$ 代表什麼呢？就是 $f(t) = t^2$

> 目標：找到最大的 m 使得 $f(m) < x$

可以列出算式 $m^2 < x$，移項之後可以得到 $m < \sqrt{x}$
所以就可以直接算出 m 就是 $\sqrt{x}$ 的整數部分

1. 如果 x 是完全平方數，答案就是 $\sqrt{x}-1$
2. 反之，答案就是 $\lfloor\sqrt{x}\rfloor$

```cpp
int ans = sqrt(x);
if(ans * ans == x) ans = ans-1;
cout << ans;
```

Subtask 4. 去 B 影城且 $x \leq 10^8$

$x \leq 10^8$ 有什麼用呢？因為 a, b 皆為正數
因此即使 $a = b = 0$，只要算算看 $f(1)$ ~ $f(10^4)$ 的值就可以考慮所有時間點的人數

```cpp
for(int i=1;i<=10000;i++) {
    int f = i * i + a * i + b;
    if(f >= x) {
        cout << i-1;
        return 0;
    }
}
```

Subtask 6. 去 B 影城且 $x \leq 10^{18}$

若 x 高達 10^{18} 就沒辦法一個一個算了
那要怎麼辦呢？可以很簡單的就看出時間越久，累積的人數就會越多

> 怎麼看出來？$a \geq 0, b \geq 0$，$f(t) = t^2 + at + b$ 當然 t 越大值就越大

也就是說 $f(t)$ 是嚴格遞增函數，所以我們就可以對他二分搜
找到最大的 m 使得 $f(m) < x$
考慮最慘的情況，$a = b = 0$，也就需要考慮到 $t = 10^9$
因此二分搜的範圍就是 $[1, 10^9]$
以下為左開右開的二分搜寫法

```cpp
int l = 0, r = 1e9+5;
while(l+1!=r) {
    int m = (l+r)>>1;
    if(m*m+m*a+b >= x) r = m;
    else l = m;
}
cout << l;
```

AC Code

```cpp
#include <bits/stdc++.h>
#define int int64_t
using namespace std;
signed main() {
    cin.tie(nullptr)->ios_base::sync_with_stdio(0);
    char q;
    int x;
    cin >> q;
    if(q == 'A') {
        int n;
        cin >> n >> x;
        vector<int> t(n);
        for(int &i : t) cin >> i;
        sort(t.begin(), t.end());
        cout << t[x-1]-1;
    }else {
        int a,b;
        cin >> x >> a >> b;
        int l = 0, r = 1e9+5;
        while(l+1!=r) {
            int m = (l+r)>>1;
            if(m*m+m*a+b >= x) r = m;
            else l = m;
        }
        cout << l;
    }
    return 0;
}
```

B. 現在正是賦權的時刻 DianSh1ng

> **難度**
> ***Easy***

> **考點**
> 輸入輸出、陣列、字串處理

Subtask 2. 所有的字串長度都是1

想像一下長度都是 1 會怎樣

- 原本是：

```
a
b
c
d
```

- 直著寫

```
dcba
```

只要依照輸入的順序反過來輸出就好了，所以恭喜你很輕鬆地拿到這個子題分

```cpp
int n;
cin >> n;
char s[n];
for(int i=0;i<n;i++) {
    cin >> s[i];
}
for(int i=n-1;i>=0;i--) {
    cout << s[i];
}
```

Subtask 3+4. 輸入的字串滿足$n = 1$

有空格跟沒空格只差在要不要用 `getline`，如果不用 `getline` 也可以用 `while` 讀

- $n = 1$ 所以呢？

```
ab cd ef
```

會變成

```
a
b

c
d

e
f
```

所以只要把讀到的字元多加一個換行輸出就好了

```cpp
string n,s;
getline(cin,n);
getline(cin,s);
for(int i=0;i<s.size();i++) {
    cout << s[i] << '\n';
}
```

啊或是你要用 `while` 輸入

```cpp
int n;
cin >> n;
string s;
while(cin >> s) {
    for(int i=0;i<s.size();i++) {
        cout << s[i] << '\n';
    }
    cout << ' ' << '\n';
}
```

Subtask 6+7. 每個字串長度可能會不一樣

- 如果不一樣要怎麼辦？

```
abc
ab
a
```

- 在前面會多出幾個空格

```
aaa
 bb
  c
```

> **其實他是二維陣列？**
> 仔細觀察就可以發現它就是二維陣列，如果很難想像的話，把空格用`#`填起來

- 原本是

```
abc
ab#
a##
```

- 會變成

```
aaa
#bb
##c
```

> **結論**
> 它其實就是把原本的陣列順時針轉 90 度

> **會遇到的問題**
> 你不知道最長的是誰，所以要找到最長的長度，然後把其他短的字串後面補上空格

AC Code

```cpp
#include <bits/stdc++.h>
#define int int64_t
using namespace std;
signed main() {
    cin.tie(nullptr)->ios_base::sync_with_stdio(0);
    string N;
    int n, maxSize = -1;
    getline(cin,N);
    n = stoi(N);
    vector<string> write(n);
    for(auto &s : write) {
        getline(cin, s);
        maxSize = max(maxSize, (int)s.size());
    }
    for(auto &s : write) {
        while(s.size() < maxSize)
            s.push_back(' ');
    }
    for(int i=0;i<maxSize;i++) {
        for(int j=n-1;j>=0;j--) {
            cout << write[j][i];
        }
        cout << '\n';
    }
    return 0;
}
```

C. 知其不可微而微之 Pseudo Differentiable

> **難度**
> ***Easy***

> **考點**
> 陣列處理

Subtask 2. $n = 0$

$n = 0$ 的意思就是 $f(x) = k$，為常數函數，微分後為零
輸出 0 就能拿到分數

```cpp
cout<<0;
```

Subtask 3. $n = 1$

$n = 1$ 的意思就是 $f(x) = ax + b$，為一次函數，其微分就是其斜率
輸出 a 就可以拿到分數

```cpp
int n,a,b;
cin >> n >> a >> b;
cout << a;
```

Subtask 4. $n = 2, p_1 = 0, p_0 = 0$

意思就是 $f(x) = ax^2 + 0x + 0 = ax^2$
微分後就是把次方往前乘之後減一，為 $2a + 0$
輸出 $(2a,0)$ 即可拿到分數

```cpp
int n,a,b,c;
cin >> n >> a >> b >> c;
cout << 2 * a << 0;
```

Subtask 5. $n = 2$

跟上個子題差不多，$f(x) = ax^2 + bx + c$
微分之後$f'(x) = 2ax + b$

```cpp
int a,b,c;
cin >> a >> b >> c;
cout << 2*a << b;
```

Subtask 6+7. $n \leq 2 \times 10^5$

子題6跟子題7其實是一樣的，只差在子題七要記得開 `long long`
就照著題目裡說的做就可以了
用一個大小為 $n + 1$ 的陣列紀錄，第 i 項就是 x^i 項的係數

```cpp
int n;
cin >> n;
// 特別處理常數函數
if(n == 0) return cout<<0,0;
vector<int> cof(n+1);
for(int &i : cof) cin >> i;
reverse(cof.begin(),cof.end());
// 係數乘到前面
for(int i=1;i<cof.size();i++)  cof[i] *= i;
reverse(cof.begin(),cof.end());
// 把常數像去掉
cof.pop_back();
for(int &i : cof) cout << i << " ";
```

D. No 1 can solve this problem

> **難度**
> ***Hell***

> **考點**
> 字串處理、數學、大數減法

> **題目來源**
> ***2024YTP*** 第一階段程式挑戰營第一題

題目在幹嘛

> 就跟提幹名稱一樣，沒有 "1" 可以解出這題
> 給你一個正整數 S，問你最少加上多少後會讓這個數字的任何位數都沒有 "1"

Subtask 2. $1 \leq S \leq 10$

直接 `if-else` 處理久好，只有 1 和 10 裡面包含 1
並且 $1+1=2, 10+10=20$

```cpp
int S;
cin >> S;
if(S == 1) cout<<1;
else if(S == 10) cout<<10;
else cout<<0;
```

Subtask 3. S 的十進制表示法中沒有 1

本來就沒有 1，答案必定為 0

```cpp
cout << 0;
```

Subtask 4. S 中最多一個 1

要注意一下，因為 $S \leq 10^{10^6}$，數字太大所以要用 `string` 讀
從整個字串裡面從左往右找找看有沒有 1，如果有的話答案就是那一位是 1，後面都是 0
比如說：2144433355343634554353 4，答案就是 10000000000000000000

```cpp
string s;
cin >> s;
for(int i=0,bool haveOne=false;i<s.size();i++) {
    if(haveOne) cout << '0';
    if(s[i] == '1') {
        cout << '1';
        haveOne = true;
    }
}
```

Subtask 5. $1 \leq S \leq 10^5$

因為保證 $S \leq 10^5$，既然S非常小，那就窮舉看看就可以了

```cpp
bool check(int x) {
    while(x) {
        if(x %10 == 1) return true;
        x /= 10;
    }
    return false;
}
```

```cpp
int n;
cin >> n;
const int iter = 1e5+5;
for(int k=0;k<=iter;k++) {
    if(!check(n+k)) {
        cout << k;
        return 0;
    }
}
```

Subtask 6. $1 \leq S \leq 10^{10^6}$

首先，前面說過了，要用 `string` 作為輸入

滿足 $S+K$ 沒有 1 的 K 有很多個，但是要最小的，因此就是可以讓他剛好進位到沒有 1 是最好的，也就是讓最左邊的 1 進位到 2

因此解法就是找到最左邊的 1，然後用那一位的 $200000..$ 減掉 S 從 1 開始的數

舉例來說：$S = 214444411244312$

原數字是 222**1**4444411244312

答案就是 000**2**0000000000000 - 000**1**4444411244312

AC Code 1

> Author : ***Zhenzhe***

```cpp
#include <bits/stdc++.h>
#define int int64_t
using namespace std;
using integar = string;
integar operator-(const integar &a, const integar &b) {
    const integar A(a.rbegin(),a.rend()), B(b.rbegin(),b.rend());
    vector<int> C(A.size());
    for(int i=0;i<C.size();i++) C[i] = (A[i]-'0') - (B[i]-'0');
    for(int i=0;i<C.size()-1;i++) {
        if(C[i] < 0) {
            C[i+1] -= 1;
            C[i] += 10;
        }
    }
    while(C.size() > 1 && C.back() == 0) C.pop_back();
    integar GET = "";
    reverse(C.begin(),C.end());
    for(auto digit : C) GET.push_back('0'+digit);
    return GET;
}
signed main() {
    cin.tie(nullptr)->ios_base::sync_with_stdio(0);
    integar s, append = "";
    cin >> s;
    bool first = 0;
    for(int i=0;i<s.size();i++) {
        if(first) append.push_back('0');
        else if(s[i] == '1') first = 1,append.push_back('2');
        else append.push_back(s[i]);
    }
    if(!first) return cout<<0,0;
    integar ans = append - s;
    return cout << ans,0;
}
```

AC Code 2

這裡有另一種方法，不是用減的，是返過來加回去

> Author : ***RainYang***

```cpp
#include<bits/stdc++.h>
using namespace std;
#define int int64_t
int32_t main(){
    string s;
    cin>>s;
    int pos=-1;
    for(int i=0;i<s.size();i++){
        if(s[i]=='1'){
            pos = i;
            break;
        }
    }
    if(pos==-1){
        cout<<0;
        return 0;
    }
    vector<int> ans;
    for(int i=s.size()-1;i>pos;i--){
        if(s[i]=='0') ans.push_back(0);
        else{
            int t = s[i]-'0';
            ans.push_back(10-t);
            if(s[i-1]=='9'){
                if(i-1<0){
                    ans.push_back(1);
                    break;
                }
                while(s[i-1]=='9'){
                    ans.push_back(0);
                    i--;
                    if(i-1<0){
                        ans.push_back(1);
                        break;
                    }
                }
                s[i-1]=s[i-1]+1;
            }
            else{
                s[i-1]=s[i-1]+1;
            }
        }
    }
    if(s[pos]=='1') ans.push_back(1);
    bool t = false;
    for(int i=ans.size()-1;i>=0;i--){
        if(ans[i]!=0) t = true;
        if(t) cout<<ans[i];
    }
}
```

E. 今日份Kahoot測驗 Today' s Kahoot

> **難度**
> ***Easy***

> **考點**
> 位元運算、進位換算

解法講很多次不想講了

自己參考比賽給的參考資料
做法就只是拿 x 一直除以 y，然後取他的餘數而已
唯一可能會遇到的問題就是超過十要轉成英文字母要怎麼辦
可以先建立一個陣列，第 k 個儲存 k 對應的符號，像是 1 對應 1，10 對應 A
簡單來說你也可以手打一個陣列

```
0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F,G,...,X,Y,Z
```

用 `vector` 會比較簡單一點

AC Code

```cpp
vector<char> dict;
int x,y;
cin >> x >> y;
if(x == 0) return cout<<0,0;
for(int i=0;i<10;i++) dict.push_back('0'+i);
for(int i=0;i<26;i++) dict.push_back('A'+i);
string cv = "";
while(x) {
    cv = dict[x%y] + cv;
    x /= y;
}
cout<<cv;
```

F. 不時可憐地以遊戲為由被社員忽略的CJ題目 Ignorance

> **難度**
> ***Medium***

> **考點**
> 邏輯、字串處理、getline

---

getline的用法和注意事項

某些子題只差在有沒有空格，解法上也只差在 `cin` 和 `getline`
使用 `getline` 可以一次讀取一整行，而 `cin` 會以空格做為分隔讀取
如果你實在不會 `getline` 你也可以用 `while` + `cin`
要特別注意的是 `cin` 和 `getline` 盡量不要同時用

如果你先 `cin` 讀 x，再用 `getline` 讀文字的話 `getline` 會讀到 x 後面的換行
所以盡量避免同時使用

- 方法一：使用 `getline` + `stoi`

```
string tempX, sentence;
getline(cin,tempX);
// stoi 可以把字串(string)型態變成整數(int)型態
int x = stoi(tempX);
getline(cin,sentence);
```

- 方法二：使用 `while` + `cin`

```
int x;
cin >> x;
string k;
while(cin >> k) {
    // do something...
}
```

如何平移 x 個字母

這題是要求向左平移 x 個字母的結果
當然可以直接拿 `char` 做加減
那當然就會遇到問題，A 左平移 2 之後是 Y
但是不直接用減的，會出問題，要怎麼辦？

- 方法一：想像成一個環
  平移終究只是在一個環裡面移動，很簡單就能發現，但重點是怎麼做？
  這就要用到數學上的餘數概念，26個循環，左平移 1 等於 左平移 $1+26$ 位
  這時候只要平移完發現跑出界線外，把他加回來就好了

```
string a,s;
getline(cin,a);
getline(cin,s);
int x = stoi(a);
for(char c : s) {
    char base = (isupper(c)? 'A' : 'a');
    if(!isalpha(c)) cout << ' ';
    else {
        int s = c - base - x;
        if(s < 0) s += 26;
        cout << (char) (base + s);
    }
}
```

- 方法二：把他多做一倍
  做出一個陣列(應該說大小寫各做一個)

```
A,B,C,....,Y,Z,A,B,C,...,Y,Z
```

這樣的話你只要從後面的 26 個去減，就可以減出你要的結果

```cpp
char dict[100];
for(int i=0;i<26;i++) dict[i] = 'A' + i;
for(int i=26;i<52;i++) dict[i] = 'A' + i;
```

G. 驚奇的心動魔法 Magic

> **難度**
> ***Hard***

> **考點**
> 數學、二分搜、中位數

最小值的思考

你數學如果好一點你基本上就把這題秒殺了

> $|a-b|$ 的意思為數線上 a, b 兩點的距離

- 當 $n=2$ 的思考
  假設 $a<b$，則 $|x-a|+|x-b|$ 的最小值？
  只要是在 $[a,b]$ 的任何數字都有最小值（包含 a, b），因為算出來都是 $|a-b|$
- 當 $n=3$ 的思考
  假設 $a<b<c$，則 $|x-a|+|x-b|+|x-c|$ 的最小值？
  當 $x=b$ 時有最小值，從 $n=2$ 可以知道
  因為 $a<b<c$，所以 $|x-a|+|x-c|=|a-c|$，為定值
  因此只要帶入的值讓 $|x-b|$ 最小就好了，那當然是 $x=b$

> 結論：$|x-a_1|+|x-a_2|+\ldots+|x-a_n|$ 的最小值發生在 $x=a$ 序列的中位數，也就是找最中間的
> 如果 n 是偶數的話，最中間那兩個隨便找一個

- 係數不為 1 的思考
  如果是 $3|x-1|+2|x-2|+|x-3|$ 怎麼辦？
  你國小肯定學過乘法 $3|x-1|=|x-1|+|x-1|+|x-1|$
  因此只要拆開找中位數就好了
  $3|x-1|+2|x-2|+|x-3|=|x-1|+|x-1|+|x-1|+|x-2|+|x-2|+|x-3|$
  ，中位數在 $[1,2]$ 之間，因此 $x=1,2$ 時都有最小值

Subtask 2+3+4. $n \leq 100$

如果你發現不了上面的事情，你還是可以窮舉看看
把所有的點都帶入找最小的，複雜度 $O(n^2)$

```cpp
int n;
cin >> n;
vector<int> p(n), m(n);
for(int &i : p) cin >> i;
for(int &i : m) cin >> i;
int ans = LLONG_MAX;
for(int i=0;i<n;i++) {
    int sum = 0;
    for(int j=0;j<n;j++) {
        sum += m[j] * abs(p[j] - p[i]);
    }
    ans = min(ans, sum);
}
cout<<ans;
```

Subtask 5+´. 所有的 M 都是 1

方法就是找中位數嘛，記得要先排序
子題五直接做排序就可以了，子題六要把係數不是 1 的都先拆開

```cpp
int n;
cin >> n;
vector<int> p(n), m(n);
for(int &i : p) cin >> i;
for(int &i : m) cin >> i;
vector<int> v;
for(int i=0;i<n;i++) {
    // 把係數不是1的拆開放到新的陣列v
    for(int j=0;j<m[i];j++) {
        v.push_back(p[i]);
    }
}
sort(v.begin(), v.end());
// 中間那個數字
int k = (v.size() + 1) / 2;
// 中位數
int x = v[k-1];
// 計算答案
int ans = 0;
for(int i=0;i<n;i++) {
    ans += m[i] * abs(x - p[i]);
}
cout<<ans;
```

Subtask 7. 無額外限制

係數最大可以到 10^5，拆開來放進一個陣列未免也太多項了
因此我們可以透過二分搜找中位數，想不到吧
先把原本的數列照 p 由小到大排序
在對數列做 M 的前綴和，就可以透過前綴和序列二分搜
找到最大的 k 使得 $prefix[k] \leq \lfloor \frac{N}{2} \rfloor$
也就是先算出係數和 N，中位數就是 $\lfloor \frac{N}{2} \rfloor$
可以使用 `lower_bound`

AC Code

```cpp
#include <bits/stdc++.h>
#define int int64_t
#define p first
#define m second
using namespace std;
signed main() {
    cin.tie(nullptr)->ios_base::sync_with_stdio(0);
    int n;
    cin >> n;
    vector<pair<int,int>> v(n);
    for(auto &i : v) cin >> i.p;
    for(auto &i : v) cin >> i.m;
    // 對 p 排序
    sort(v.begin(),v.end());
    vector<int> pref(n);
    pref[0] = v[0].m;
    // 計算前綴和
    for(int i=1;i<n;i++) pref[i] = pref[i-1] + v[i].m;
    // 找到中位數
    int k = (pref.back()+1)/2;
    auto f = lower_bound(pref.begin(), pref.end(), k) - pref.begin();
    int x = v[f].p, ans = 0;
    // 代入計算答案
    for(int i=0;i<n;i++) ans += v[i].m * abs(x - v[i].p);
    return cout<<ans,0;
}
```

H. 多麼虎頭蛇尾的結局 Goodbye

> **難度**
> ***Very Easy***

> **考點**
> 上課記得要聽

AC Code

懶得講自己看

```cpp
#include <bits/stdc++.h>
using namespace std;
int main() {
    string ans[16] = {
        "mingyee",
        "Goodbye!\\><\/",
        "hello, world",
        "04/30",
        "zhenzhe",
        "hanks2017",
        "sh1ng",
        "citrc",
        "2",
        "4",
        "7",
        "26",
        "3",
        "224",
        "4",
        "kita ikuyo"
    };
    int q;
    cin>>q;
    return std::cout<<ans[q],0;
}
```

I. 巨震大師 MatrixMaster

> **難度**
> ***Hard***

> **考點**
> 快速冪、數學、struct

就是上課講的快速冪，一模一樣

> ~~不，完全不一樣~~

Subtask 1. 求 1^n

```cpp
cout << 1;
```

Subtask 2. 求 a^n

- 遞迴版

```cpp
const int mod = 1e9+7;
long long fpow(long long a,long long n) {
    if(n == 0) return 1;
    if(n % 2) return a * fpow(a,n-1) % mod;
    long long half = fpow(a,n/2);
    return half * half % mod;
}
```

- 迴圈版

```cpp
long long ans = 1;
while(n) {
    if(n & 1) ans *= a, ans %= mod;
    a *= a;
    a %= mod;
    n >>= 1;
}
```

Subtask 7+8. $n \leq 10^5$

用 `for` 迴圈跑一遍就會了

```cpp
int a,b,c,d,n;
for(int i=0;i<n;i++) {
    // a = ...
    // b = ...
    // c = ...
    // d = ...
}
```

Subtask 3+4+5+9. 無額外限制

先用 `strucr` 自定義一個乘法
定義 `struct` 為 $(p+q\sqrt{c})+(r+s\sqrt{c})i$

```cpp
struct Complex{
    int p,q,r,s;
    int c;
};
```

再定義乘法的方式

```cpp
Complex times(Complex A, Complex B) {
    Complex ans;
    ans.p = (A.p*B.p+A.q*B.q*c-A.r*B.r-A.s*B.s*c) % mod;
    ans.q = (A.p*B.q+A.q*B.p-A.r*B.s-A.s*B.r) % mod;
    ans.r = (A.p*B.r+A.q*B.s*c+A.r*B.p+A.s*B.q*c) % mod;
    ans.s = (A.p*B.s+A.q*B.r+A.r*B.q+A.s*B.p) % mod;
    ans.c = A.c;
}
```

套回去快速冪，把資料型態改掉

```cpp
Complex fpow(Complex a,long long n) {
    if(n == 1) return a;
    if(n % 2) return times(a,fpow(a,n-1));
    long long half = fpow(a,n/2);
    return times(half,half);
}
```

> 不同子題的解法就是定義不同的 struct
> 比如說只要求 $(a+b\sqrt{c})$ 就只要定義 (a,b) 兩個就好
> 乘法定義為 $(p+q\sqrt{c})(r+s\sqrt{c}) = (pr+qsc)+(qr+ps)\sqrt{c}$
> 也就是輸入 $(p,q),(r,s)$，會回傳 $(pr+qsc, qr+ps)$
> 當然記得要取餘數

矩陣解法

> 你猜為什麼這題叫做巨震大師？~~那肯定要用矩陣解啊~~

轉移矩陣如下：

$$\begin{bmatrix} p & qr & -s & 0 \\ q & p & 0 & -s \\ s & 0 & p & qr \\ 0 & s & q & p \end{bmatrix}$$

Code 僅供參考

```cpp
#include <bits/stdc++.h>
#define int int64_t
using namespace std;
static constexpr int mod = 1e9+7;
using matrix = vector<vector<int>>;
matrix operator*(const matrix &A, const matrix &B) {
    matrix C(A.size(), vector<int> (B[0].size(),0));
    for(int k=0;k<B.size();k++) {
        for(int i=0;i<A.size();i++) {
            for(int j=0;j<B[0].size();j++) {
                C[i][j] += A[i][k] * B[k][j] % mod;
                C[i][j] %= mod;
            }
        }
    }
    return C;
}
signed main() {
    cin.tie(nullptr)->ios_base::sync_with_stdio(0);
    int n,a,b,c,d;
    cin >> n >> a >> b >> c >> d;
    matrix T = {
        {a,b*c,0-d,0},
        {b,a,0,0-d},
        {d,0,a,b*c},
        {0,d,b,a}
    };
    matrix S = {
        {1},
        {0},
        {0},
        {0}
    };
    while(n) {
        if(n & 1) S = T * S;
        n >>= 1;
        T = T * T;
    }
    for(int i=0;i<4;i++) {
        cout << S[i][0] << " \n"[i==3];
    }
}
```

J. 計畫C Plan Cr

> **難度**
> ***Easy***

> **考點**
> 浮點數運算、數學

一元二次方程式公式解還需要我教你嗎？

Subtask 1. $b^2 - 4ac < 0$

很明顯無實數解，輸出就有分

```cpp
cout << "no real root";
```

Subtask 2. 兩根必定為整數

兩根為 α, β，方程式為 $x^2 - (\alpha + \beta) + \alpha\beta = 0$
你可以解聯立方程式，也可以直接用公式解，因為是整數，所以不用處理小數點的問題

Subtask 3. 無額外限制

直接帶公式解，並且因為 $a > 0$，因此取正必定較大
記得開 `double` ，開 `float` 精度不夠

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```cpp
#include <bits/stdc++.h>
using namespace std;
signed main() {
    cin.tie(nullptr)->ios_base::sync_with_stdio(0);
    int a,b,c;
    cin >> a >> b >> c;
    int D = b*b-4*a*c;
    if(D < 0) return cout<<"no real root\n",0;
    double d = sqrt(D), aa = a * 2.0;
    cout << fixed << setprecision(15) << (0-b+d) / aa << '\n';
    cout << fixed << setprecision(15) << (0-b-d) / aa << '\n';
}
```