1 pA

給定三個點求一元二次方程式係數
策略: 因為題目說了所有數字 ≤ 64, 所以直接枚舉就行了

```cpp
#include <cstdio>
using namespace std;

bool check(int a, int b, int c, int x, int y){
    return y == a * x * x + b * x + c;
}

void Solve(){
    int x[3], y[3];
    for(int i = 0 ; i < 3 ; i++)
        scanf("%d%d", &x[i], &y[i]);
    for(int i = -64 ; i <= 64 ; i++)
        for(int j = -64 ; j <= 64 ; j++)
            for(int k = -64 ; k <= 64 ; k++){
                bool ok = true;
                for(int l = 0 ; l < 3 ; l++)
                    ok &= check(i, j, k, x[l], y[l]);
                if(ok)
                    printf("%d %d %d\n", i, j, k);
            }

}

int main(){
    int n;
    scanf("%d", &n);
    while(n--)
        Solve();
    return 0;
}
```

2 pC

給定一棵樹需要將樹上的點塗色每個沒塗色的點都需要至少和兩個塗色的點相鄰問至少要塗多少點

策略: 動態規劃,dp[n][3] 對於每個節點有三種狀態, 塗色, 不塗色且至少一個 child 有塗色, 不塗色且至少有兩個 child 塗色。

```cpp
#include <iostream>
#include <vector>
using namespace std;

const int INF = 1024;

int dp[128][3];
vector<int> ed[128];

void dfs(int x){
    dp[x][1] = dp[x][2] = INF;
    dp[x][0] = 1;
    if(ed[x].size() == 0){
    } else if(ed[x].size() == 1) {
        int u = ed[x][0];
        dfs(u);
        dp[x][0] += min(dp[u][0], min(dp[u][1], dp[u][2]));
        dp[x][1] = dp[u][0];
    } else {
        int cnt = 0, total = 0, ma = INF, mb = INF;
        for(int i = 0 ; i < (int)ed[x].size() ; i++){
            int u = ed[x][i];
            dfs(u);
            dp[x][0] += min(dp[u][0], min(dp[u][1], dp[u][2]));
            if(dp[u][0] <= dp[u][2]){
                cnt++;
                total += dp[u][0];
            } else {
                total += dp[u][2];
                int tmp = dp[u][0] - dp[u][2];
                if(tmp <= ma){
                    mb = ma;
                    ma = tmp;
                } else if(tmp <= mb){
                    mb = tmp;
                }
            }
        }
        dp[x][1] = dp[x][2] = total;
        if(cnt == 0){
            dp[x][1] += ma;
            dp[x][2] += ma + mb;
        } else if(cnt == 1){
            dp[x][2] += ma;
        }
    }
    //printf("%d: %d %d %d\n", x, dp[x][0], dp[x][1], dp[x][2]);
}

void Solve(){
    int n;
    scanf("%d", &n);
    int t;
    for(int i = 1 ; i <= n ; i++){
        ed[i].clear();
```

```
56          scanf("%d", &t);
57          if(i == 1) continue;
58          ed[t].push_back(i);
59      }
60      dfs(1);
61      printf("%d\n", min(dp[1][0], dp[1][2]));
62  }
63
64  int main(){
65      int n;
66      scanf("%d", &n);
67      while(n--)
68          Solve();
69      return 0;
70  }
```

3 pD

約瑟夫殺人問題，隔 Fibonacci 數列殺人，搭配遇到質數轉向
策略: 先將 Fibonacci 跟質數表建好，接著照著走就行了

```cpp
#include <cstdio>
#include <vector>
using namespace std;
typedef long long ll;
ll f[36];
bool p[36];
void Solve(){
    int n;
    scanf("%d", &n);
    bool exist[36];
    fill(exist, exist+sizeof(exist), 1);
    vector<int> v;
    for(int i = 1 ; i <= n ; i++)
        v.push_back(i);
    int now = 1, remain, d = 1;
    for(int i = 0 ; i < n - 1 ; i++){
        int tmp = n - i;
        remain = f[i] % tmp;
        if(remain == 0) remain += tmp;
        while(remain != 1){
            now += d;
            if(now < 1) now = n;
            if(now > n) now = 1;
            if(exist[now])
                remain--;
        }
        if(p[now]) d *= -1;
        exist[now] = 0;
        while(!exist[now]){
            now += d;
            if(now < 1) now = n;
            if(now > n) now = 1;
        }
    }
    for(int i = 1 ; i <= n ;i++)
        if(exist[i])
            printf("%d\n", i);
}
int main(){
    f[0] = f[1] = 1;
    for(int i = 2 ; i < 35 ; i++)
        f[i] = f[i-1] + f[i-2];
    p[2] = p[3] = p[5] = p[7] = p[11] = p[13] = p[17] = p[19] = p[23] = p[29] = p[31] = 1;
    int n;
    scanf("%d", &n);
    while(n--)
        Solve();
    return 0;
}
```

4 pE

給定一張圖問你有幾條路可以從 s 走道 t
策略: 直接 DFS

```cpp
#include <cstdio>
#include <vector>
using namespace std;
int n, m, s, t;
int ans;
vector<int> ed[16];
bool use[16];
void dfs(int now){
    if(now == t){
        ans++;
        return;
    }
    use[now] = true;
    for(int i = 0 ; i < (int)ed[now].size() ; i++){
        if(!use[ed[now][i]])
            dfs(ed[now][i]);
    }
    use[now] = false;
}
void Solve(){
    scanf("%d%d%d%d", &n, &m, &s, &t);
    for(int i = 1 ; i <= n ; i++){
        int tmp, p;
        scanf("%d", &tmp);
        ed[i].clear();
        for(int j = 0 ; j < tmp ; j++){
            scanf("%d", &p);
            ed[i].push_back(p);
        }
    }
    ans = 0;
    dfs(s);
    printf("%d\n", ans);
}

int main(){
    int n;
    scanf("%d", &n);
    while(n--)
        Solve();
    return 0;
}
```

5 pF

給定 n 組 x, y 求出 k, 其中 $y = k + a_1 * x + a_2 * x^1 ... a_{n-1} * x^{n-1}$
策略: 直接做高斯消去即可

```cpp
#include <vector>
#include <cstdio>
using namespace std;
#define MAX 1048576
typedef long long ll;
int p;
int rev_mod[MAX];
int mpow(ll a, int b){
    ll re = 1;
    while(b){
        if(b&1)
            re = a * re % p;
        a = a * a % p;
        b >>= 1;
    }
    return re;
}
class Matrix{
public:
    vector<vector<ll> > D;
    int R, C;
    Matrix(): R(0), C(0) {}

    Matrix(int r, int c): R(r), C(c) {
        D = vector< vector<ll> > (R);
        for(int i  = 0 ; i < R ; i++)
            D[i] = vector<ll>(C);
    }

    ll& at(const int &rhs1, const int &rhs2) {
        return D[rhs1][rhs2];
    }

    const ll& at(const int &rhs1, const int &rhs2) const {
        return D[rhs1][rhs2];
    }

    vector<ll> Solve() {
        vector<ll> res(R);
        for(int i = 0 ; i < R ; i++){
            for(int j = i ; j < R ; j++){
                if(at(j, i)){
                    swap(D[i], D[j]);
                    break;
                }
            }
            for(int j = 0 ; j < R ; j++){
                if(i == j) continue;
                int t = at(j, i) * rev_mod[at(i, i)] % p;
                for(int k = i ; k < C ; k++){
                    at(j, k) -= at(i, k) * t % p;
                    at(j, k) = (at(j, k) + p) % p;
                }

            }
        }
```

```
57          return res;
58      }
59  };
60  int main(){
61      scanf("%d", &p);
62      for(int i = 1 ; i < p ; i++){
63          rev_mod[i] = mpow(i, p-2);
64      }
65      int t;
66      while(scanf("%d", &t), t){
67          Matrix m(t, t + 1);
68          for(int i = 0 ; i < t ; i++){
69              int x, y;
70              scanf("%d%d", &x, &y);
71              m.at(i, 0) = 1;
72              for(int j = 1 ; j < t ; j++){
73                  m.at(i, j) = m.at(i, j-1) * x % p;
74              }
75              m.at(i, t) = y;
76          }
77          m.Solve();
78          printf("%lld\n", m.at(0, t));
79      }
80      return 0;
81  }
```

6 pH

給定一棵樹將點兩兩圈起問最多可以有幾個 pair 以及有多少點是關鍵點 (不管哪種圈法一定會被圈到)

策略: 如果孩子裡面有沒配對到的則自己一定是關鍵點 (此數量也會等於 pair 的數量)

再把所有關建點丟進 queue 裡面如果關鍵點只有連到一個不是關鍵點的點則被連到的點也是關鍵點

```cpp
#include <vector>
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <queue>
using namespace std;
#define MAX 131072
vector<int> ed[MAX];
bool crit[MAX];
int ans_pair;

void dfs(int x, int p = -1){
    for(int i = 0 ; i < (int)ed[x].size() ; i++){
        int &u = ed[x][i];
        if(u == p) continue;
        dfs(u, x);
        if(crit[u] == 0) crit[x] = 1;
    }
    if(crit[x]) ans_pair++;
}

void Solve(){
    ans_pair = 0;
    memset(crit, 0, sizeof(crit));
    int n;
    scanf("%d", &n);
    for(int i = 0 ; i < n ; i++){
        ed[i].clear();
    }
    int x;
    for(int i = 1 ; i < n ; i++){
        scanf("%d", &x);
        ed[x].push_back(i);
        ed[i].push_back(x);
    }
    dfs(0);
    vector<int> ans;
    queue<int> q;
    for(int i = 0 ; i < n ; i++)
        if(crit[i]){
            ans.push_back(i);
        }
    for(int i = 0 ; i < (int)ans.size() ; i++){
        int num = 0, tmp;
        for(int j = 0 ; j < (int)ed[i].size() ; j++){
            int &u = ed[i][j];
            if(crit[u] == 0){
                num++;
                tmp = u;
            }
        }
        if(num == 1){
            crit[tmp] = 1;
            ans.push_back(tmp);
```

```
55          }
56      }
57      sort(ans.begin(), ans.end());
58      printf("%d %d\n", ans_pair, (int)ans.size());
59      for(int i = 0 ; i < min((int)ans.size(), 3) ; i++)
60          printf("%d%c", ans[(int)ans.size()-i-1], i == min((int)ans.size(), 3) - 1 ? '\n' :
                ' ');
61  }
62
63  int main(){
64      int n;
65      scanf("%d", &n);
66      while(n--)
67          Solve();
68      return 0;
69  }
```

7 pK

給定兩排數字每排六個一位數整數任一挑一些數字做任意排列問兩邊有多少個對方沒有的數字
策略: 用 next_permutation 搭配 set 實作即可

```cpp
#include <cstdio>
#include <algorithm>
#include <set>
using namespace std;

void Solve(){
    int a[2][6], ans[2];
    set<int> s[2];
    for(int i = 0 ; i < 2 ; i++){
        for(int j = 0 ; j < 6 ; j++)
            scanf("%d", &a[i][j]);
        sort(a[i], a[i]+6);
        do{
            int now = 0;
            for(int j = 0 ; j < 6 ; j++){
                now = now * 10 + a[i][j];
                s[i].insert(now);
            }
        }while(next_permutation(a[i], a[i]+6));
    }
    for(int i = 0 ; i < 2 ; i++){
        ans[i] = s[i].size();
        for(set<int>::iterator x = s[i].begin() ; x != s[i].end() ; ++x)
            if(s[i^1].count(*x))
                ans[i]--;
    }
    printf("%d %d\n", ans[0], ans[1]);

}

int main(){
    int n;
    scanf("%d", &n);
    while(n--)
        Solve();
    return 0;
}
```