Dynamic Programming

- **Ad hoc**
  - UVA 825 - Walking on the Safe Side
  - UVA 986 - How Many?
  - TIMUS 1017 - Staircases
  - TIMUS 1018 - Binary Apple Tree
  - UVA 1231 - ACORN
  - UVA 1239 - Greatest K-Palindrome Substring
  - TIMUS 1658 - Sum of Digits
  - UVA 10444 - Multi-peg Towers of Hanoi
  - UVA 11375 - Matches
  - UVA 11703 - sqrt log sin
- **Edit Distance**
  - UVA 1207 - AGTC
  - UVA 10739 - String to Palindrome
- **Integer partition**
  - UVA 907 - Winterim Backpacking Trip
- **Josephus Problem**
  - UVA 10015 - Joseph's Cousin
- **Knapsack**
  - **Binary Knapsack**
    - UVA 10930 - A-Sequence
    - UVA 11658 - Best Coalitions
  - **Counting Knapsack**
    - UVA 1213 - Sum of Different Primes
  - **Infinite Items Knapsack**
    - UVA 1158 - CubesSquared
- **Longest Common Subsequence**
  - UVA 10066 - The Twin Towers
  - UVA 10192 - Vacation
  - UVA 10723 - Cyborg Genes
  - UVA 11151 - Longest Palindrome
  - UVA 12147 - DNA Sequences
- **Longest Common Substring**
  - UVA 1223 - Editor
- **Longest Increasing Subsequence**
  - UVA 473 - Raucous Rockers
  - UVA 10051 - Tower of Cubes
  - UVA 10154 - Weights and Measures
  - UVA 10635 - Prince and Princess
  - UVA 11003 - Boxes
- **Matrix Multiplication**
  - UVA 10003 - Cutting Sticks
  - UVA 10891 - Game of Sum
- **Maximum Sum Contiguous Subsequence**
  - UVA 10684 - The Jackpot
  - UVA 11059 - Maximum Product
- **Maximum Sum Sub-rectangle**
  - UVA 10827 - Maximum sum on a torus
- **Minimax**

- UVA 12484 - Cards

- **Optimal Search Tree**
  - UVA 10304 - Optimal Binary Search Tree

Graphs

- **2-SAT**
  - UVA 10319 - Manhattan
  - UVA 11294 - Wedding
- **Bipartite Matching**
  - UVA 11159 - Factors and Multiples
  - UVA 12159 - Gun Fight
  - **Konig Theorem**
    - UVA 11419 - SAM I AM
    - UVA 12168 - Cat vs. Dog
- **DFS**
  - UVA 273 - Jack Straws
  - UVA 1197 - The Suspects
  - UVA 1216 - The Bug Sensor Problem
  - UVA 1220 - Party at Hali-Bula
  - UVA 10113 - Exchange Rates
  - UVA 10243 - Fire! Fire! Fire!
  - UVA 10259 - Hippity Hopscotch
  - UVA 12186 - Another Crisis
- **Finding Articulation Points**
  - UVA 315 - Network
  - UVA 10199 - Tourist Guide
- **Finding Bridges**
  - UVA 610 - Street Directions
  - UVA 796 - Critical Links
  - UVA 12363 - Hedge Mazes
- **Flood Fill**
  - UVA 11110 - Equidivisions
  - UVA 11518 - Dominos 2
- **Job Scheduling**
  - UVA 1205 - Color a Tree
- **Markov Chain**
  - UVA 12487 - Midnight Cowboy
- **Maximum Flow**
  - **Ford-Fulkerson**
    - UVA 820 - Internet Bandwidth
    - UVA 10092 - The Problem with the Problem Setter
    - UVA 10480 - Sabotage
    - UVA 10511 - Councilling
  - **Min Cost**
    - **Cycle Canceling**
      - UVA 10594 - Data Flow
      - UVA 10746 - Crime Wave - The Sequel
- **Minimum Spanning Tree**
  - **Kruskal**
    - UVA 1265 - Tour Belt
    - UVA 10462 - Is There A Second Way Left?
    - UVA 11857 - Driving Range

    - **Prim**
        - UVA 908 - Re-connecting Computer Sites
        - UVA 1208 - Oreon
        - UVA 1235 - Anti Brute Force Lock
        - **Priority Queue**
            - UVA 1174 - IP-TV
            - UVA 1234 - RACING
            - UVA 10397 - Connect the Campus
            - UVA 11631 - Dark roads
            - UVA 11733 - Airports
            - UVA 11747 - Heavy Cycle Edges

- **Shortest Path**
    - **Bellman Ford**
        - UVA 10557 - XYZZY
    - **BFS**
        - UVA 298 - Race Tracks
        - UVA 314 - Robot
        - UVA 321 - The New Villa
        - UVA 627 - The Net
        - UVA 652 - Eight
        - UVA 1251 - Repeated Substitution with Sed
        - UVA 10044 - Erdos Number
        - UVA 12101 - Prime Path
        - UVA 12135 - Switch Bulbs
        - UVA 12160 - Unlock the Lock
    - **Dijkstra**
        - UVA 929 - Number Maze
        - UVA 1247 - Interstar Transport
        - UVA 10389 - Subway
        - UVA 10986 - Sending email
        - UVA 11280 - Flying to Fredericton
        - UVA 11833 - Route Change
        - UVA 12144 - Almost Shortest Path
    - **Floyd-Warshall**
        - UVA 1056 - Degrees of Separation
        - UVA 1233 - USHER
        - UVA 10278 - Fire Station
        - UVA 10724 - Road Construction
        - UVA 10793 - The Orc Attack
        - UVA 12179 - Randomly-priced Tickets

- **Strongly Connected Components**
    - UVA 1229 - Sub-Dictionary
    - UVA 11709 - Trust Groups
    - UVA 11838 - Come and Go

- **Topological Sorting**
    - UVA 1263 - Mines
    - UVA 11686 - Pick up Sticks
    - UVA 11770 - Lighting Away

- **Tree Isomorphism**
    - UVA 12489 - Combating cancer

Math

- **Big Integer**
    - UVA 424 - Integer Inquiry

- **Extended Euclid**
    - UVA 10090 - Marbles
    - UVA 10104 - Euclid Problem

- **GCD**
    - UVA 12184 - Transcribed Books

- **Geometry**
  - TIMUS 1020 - Rope
  - UVA 12194 - Isosceles Triangles
  - UVA 12300 - Smallest Regular Polygon
  - **3D Line Detection**
    - TIMUS 1422 - Fireflies
  - **Convex Hull**
    - UVA 109 - SCUD Busters
    - **Monotone Chain**
      - UVA 361 - Cops and Robbers
      - UVA 811 - The Fortified Forest
      - TIMUS 1185 - Wall
      - UVA 10065 - Useless Tile Packers
      - UVA 10652 - Board Wrapping
      - UVA 11096 - Nails
  - **Enclosing Circle**
    - TIMUS 1185 - Wall
    - TIMUS 1332 - Genie Bomber
  - **Great-Circle Distance**
    - TIMUS 1030 - Titanic
    - UVA 10316 - Airline Hub
  - **Mirror**
    - TIMUS 1258 - Pool
  - **Point Sort**
    - UVA 11626 - Convex Hull
  - **Point to Line**
    - UVA 12483 - Toboggan of Marbles
  - **Segment Rotation**
    - TIMUS 1373 - Pictura ex Machina
  - **Segments Angle**
    - TIMUS 1578 - Mammoth Hunt
  - **Square Distance**
    - TIMUS 1111 - Squares
- **Prime Factorization**
  - UVA 12137 - Puzzles of Triangles
  - **Euler's Totient**
    - UVA 12493 - Stars
- **Probability**
  - UVA 11762 - Race to 1
- **Sieve**
  - UVA 1246 - Find Terrorists

Misc

- **Ad hoc**
  - UVA 136 - Ugly Numbers
  - UVA 160 - Factors and Factorials
  - UVA 458 - The Decoder
  - UVA 494 - Kindergarten Counting Game
  - UVA 573 - The Snail

    - UVA 579 - ClockHands
    - UVA 579 - ClockHands
    - UVA 591 - Box of Bricks
    - UVA 10018 - Reverse and Add
    - UVA 10035 - Primary Arithmetic
    - UVA 10189 - Minesweeper
    - UVA 10300 - Ecological Premium
    - UVA 10694 - f91
    - UVA 10783 - Odd Sum
    - UVA 11494 - Queen
    - UVA 11597 - Spanning Subtree
    - UVA 12148 - Electricity
    - UVA 12155 - ASCII Diamondi
    - UVA 12195 - Jingle Composing
    - UVA 12196 - Klingon Levels
    - UVA 12482 - Short Story Competition
    - UVA 12485 - Perfect Choir
    - UVA 12488 - Start Grid
    - UVA 12490 - Integral
    - UVA 12492 - Rubik Cycle
- **Binary Manipulation**
    - UVA 11532 - Simple Adjacency Maximization
- **Binary Search**
    - UVA 1215 - String Cutting
    - UVA 12190 - Electric Bill
    - UVA 12192 - Grapevine
    - UVA 12486 - Space Elevator
- **Fenwick Tree**
    - UVA 11423 - Cache Simulator
    - UVA 11525 - Permutation
    - UVA 11610 - Reverse Prime
    - UVA 12086 - Potentiometers
    - UVA 12365 - Jupiter Atacks!
    - **2D**
        - SPOJ NKMOBILE - IOI01 Mobiles
- **Greed**
    - UVA 12172 - Matchsticks
- **Linked List**
    - UVA 245 - Uncompress
- **Permutation Cycle**
    - UVA 1016 - Silly Sort
    - UVA 12103 - Leonardo's Notebook
- **Priority queue**
    - UVA 1203 - Argus
- **Segment Tree**
    - UVA 1232 - SKYLINE
    - **2D**
        - UVA 11297 - Census
    - **Lazy Propagation**
        - UVA 11402 - Ahoy, Pirates!
        - SPOJ BR HOMEM - Homem, Elefante e Rato
    - **Range Maximum Query**
        - UVA 11235 - Frequent Values
- **Sort**
    - UVA 11157 - Dynamic Frog
    - UVA 12189 - Dinner Hall
- **STL map**

- UVA 119 - Greedy Gift Givers
- UVA 902 - Password Search
- UVA 10420 - List of Conquests
- UVA 11629 - Ballot evaluation
- UVA 12491 - Words

- **String Matching**
  - **KMP**
    - UVA 10298 - Power Strings
    - **2D**
      - UVA 422 - Word-Search Wonder
    - **Suffix-Prefix**
      - UVA 11475 - Extend to Palindrome
      - UVA 11576 - Scrolling Sign
  - **Suffix Array**
    - **Circular**
      - UVA 719 - Glass Beads
    - **Longest Common Prefix**
      - UVA 760 - DNA Sequencing
      - UVA 11512 - GATTACA
      - UVA 12361 - File Retrieval
  - **Trie**
    - UVA 11590 - Prefix Lookup
    - UVA 12506 - Shortest Names
- **String parsing**
  - UVA 1200 - A DP Problem
- **Union-Find**
  - UVA 10158 - War
  - UVA 11503 - Virtual Friends
  - UVA 11966 - Galactic Bonding

uva/109.cpp

```cpp
//109
//SCUD Busters
//Math;Geometry;Convex Hull
#include <iostream>
#include <cmath>
#include <iomanip>
#include <algorithm>
using namespace std;

struct Point {
    int x, y;

    Point() {}
    Point(int x, int y) : x(x), y(y) {}

    bool left(Point& a, Point& b) {
        return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x) < 0;
    }

    static bool lesserX(Point& p1, Point& p2) {
        return p1.x < p2.x;
    }

    bool operator ==(const Point& p) const {
        return this->x == p.x && this->y == p.y;
    }

};

double area(Point* A, int a) {
    double area = 0;
    for(int i=0; i<a; i++) {
        int j = (i+1)%a;
        area += (A[i].x + A[j].x) * (A[i].y - A[j].y);
    }
    return area / 2;
}

int convexHull(Point* P, int n, Point* S) {
    int m=0;
    S[m++] = *min_element(P, P+n, Point::lesserX);
    while(true) {
        Point cand = S[m-1];
        for(int j=0; j<n; j++)
            if (cand==S[m-1] or P[j].left(S[m-1], cand))
                cand = P[j];

        if (cand == S[0]) break;
        S[m++] = cand;
    }
    return m;
}

//assumes convex polygon, ordered clockwise
//convex hull output is just that
bool checkInside(Point* P, int n, Point v) {
    for(int i=1; i<=n; i++)
        if (v.left(P[i-1], P[i%n]))
            return false;

    return true;
}


Point P[30][250], S[30][250];
int N[30], M[30];
bool V[30];

int main() {
    int n, k=0;
    while(cin >> n, n!=-1) {
        N[k] = n;
        for(int i=0; i<n; i++) {
            int x,y; cin >> x >> y;
            P[k][i] = Point(x,y);
        }

        M[k] = convexHull(P[k], n, S[k]);
```

```
79
80          k++;
81      }
82
83      int x, y;
84      while(cin >> x >> y) {
85          for(int i=0; i<k; i++)
86              if (checkInside(S[i], M[i], Point(x,y)))
87                  V[i] = true;
88      }
89
90      double total = 0;
91      for(int i=0; i<k; i++)
92          if (V[i])
93              total += area(S[i], M[i]);
94
95      cout << fixed << setprecision(2) << total << endl;
96  }
```

uva/119.cpp

```
1   //119
2   //Greedy Gift Givers
3   //Misc;STL map
4   #include <iostream>
5   #include <map>
6   #include <vector>
7   #include <string>
8   using namespace std;
9
10  map<string, int> M;
11  vector<string> V;
12
13  int main() {
14      int n, t=0;
15      while(cin >> n) {
16          M.clear(); V.clear();
17          if (t++) cout << endl;
18
19          for(int i=0; i<n; i++) {
20              string s;
21              cin >> s;
22              M[s] = 0;
23              V.push_back(s);
24          }
25
26          for(int i=0; i<n; i++) {
27              string a; int g, m;
28              cin >> a >> g >> m;
29              if (!m) continue;
30              g /= m;
31
32              for(int j=0; j<m; j++) {
33                  string b;
34                  cin >> b;
35                  M[b] += g;
36                  M[a] -= g;
37              }
38          }
39
40          for(int i=0; i<n; i++)
41              cout << V[i] << " " << M[V[i]] << endl;
42      }
43
44  }
```

uva/136.cpp

```
1   //136
2   //Ugly Numbers
3   //Misc;Ad hoc
4   #include <iostream>
5   #include <queue>
6   #define ull unsigned long long
7   using namespace std;
8
9   struct Number {
10      ull n, p;
11      Number(ull n, ull p) : n(n), p(p) {}
```

```
    inline const bool operator < (const Number& that) const {
        return n > that.n;
    }
};

priority_queue<Number> Q;
int main() {
    Q.push(Number(1,1));
    for(int i=1; i<1500; i++) {
        Number last = Q.top(); Q.pop();
        if (last.p <= 2) Q.push(Number(last.n*2, 2));
        if (last.p <= 3) Q.push(Number(last.n*3, 3));
        if (last.p <= 5) Q.push(Number(last.n*5, 5));
    }

    cout << "The 1500'th ugly number is " << Q.top().n << "." << endl;

    return 0;
}
```

uva/160.cpp

```
//160
//Factors and Factorials
//Misc;Ad hoc
#include <iostream>
#include <iomanip>
using namespace std;

int W[] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97 }, wn = 2
int T[101][25];

int main() {
    for(int i=2; i<=100; i++) {
        int p = i;
        for(int j=0; j<wn; j++)
            T[i][j] = T[i-1][j];

        for(int j=0; j<wn && p>1; j++) {
            while(p%W[j]==0) {
                p/=W[j];
                T[i][j]++;
            }
        }
    }

    int n;
    while(cin >> n, n) {
        cout <<  setw(3) << right << n << "! =";
        int count = 0;
        for(int i=0; i<wn; i++) {
            if (T[n][i] == 0) break;
            if (++count > 15) {
                count = 1;
                cout << endl << "      ";
            }
            cout << setw(3) << right << T[n][i];
        }
        cout << endl;
    }

    return 0;
}
```

uva/245.cpp

```
//245
//Uncompress
//Misc;Linked List
#include <iostream>
#include <sstream>
#include <string>
#include <climits>
#include <cstring>
#include <cstdio>
#include <list>
#define MAX 1000
using namespace std;

```

```
list<string> W;
stringstream sin;
int curnum=0;
bool word=false, number=false;

void finishWord() {
    W.push_back(sin.str());
    sin.str("");
    word = false;
}

void finishNumber() {
    list<string>::iterator it = W.end();
    while(curnum--)
        it--;

    cout << *it;
    W.push_back(*it);
    W.erase(it);

    curnum = 0;
    number = false;
}

int main() {
    string s;

    while(getline(cin, s), s!="0") {
        for(int i=0; i<s.size(); i++) {
            char c = s[i];
            if (c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z') {
                sin << c;
                word = true;
            } else if (word) finishWord();

            if (c >= '0' && c <= '9') {
                curnum *= 10; curnum += c-'0';
                number = true;
            } else if (number) finishNumber();

            if (!number)
                cout << c;
        }
        if (word) finishWord();
        if (number) finishNumber();

        cout << endl;
    }
}
```

uva/273.cpp

```
//273
//Jack Straws
//Graphs;DFS
#include <iostream>
#include <cstring>
#define MAX 100002
using namespace std;

static bool segment(int xi, int yi, int xj, int yj,
                    int xk, int yk) {
  return (xi <= xk || xj <= xk) && (xk <= xi || xk <= xj) &&
         (yi <= yk || yj <= yk) && (yk <= yi || yk <= yj);
}

static char direction(int xi, int yi, int xj, int yj,
                      int xk, int yk) {
  int a = (xk - xi) * (yj - yi);
  int b = (xj - xi) * (yk - yi);
  return a < b ? -1 : a > b ? 1 : 0;
}

bool intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {
  char d1 = direction(x3, y3, x4, y4, x1, y1);
  char d2 = direction(x3, y3, x4, y4, x2, y2);
  char d3 = direction(x1, y1, x2, y2, x3, y3);
  char d4 = direction(x1, y1, x2, y2, x4, y4);
  return (((d1 > 0 && d2 < 0) || (d1 < 0 && d2 > 0)) &&
          ((d3 > 0 && d4 < 0) || (d3 < 0 && d4 > 0))) ||
         (d1 == 0 && segment(x3, y3, x4, y4, x1, y1)) ||
```

```
         (d2 == 0 && segment(x3, y3, x4, y4, x2, y2)) ||
         (d3 == 0 && segment(x1, y1, x2, y2, x3, y3)) ||
         (d4 == 0 && segment(x1, y1, x2, y2, x4, y4));
}

int G[20][20], V[20], A[20], B[20], C[20], D[20], n;

int dfs(int v, int comp) {
    V[v] = comp;
    for(int i=1; i<=n; i++)
        if (!V[i] && G[v][i])
            dfs(i, comp);
}

int main() {
    int t; cin >> t; t=0;
    while(cin >> n) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        for(int i=1; i<=n; i++) {
            cin >> A[i] >> B[i] >> C[i] >> D[i];
            for(int j=1;j<i; j++)
                G[i][j] = G[j][i] = intersect(A[i], B[i], C[i], D[i], A[j], B[j], C[j], D[j]);
        }

        int compn = 0;
        for(int i=1; i<=n; i++)
            if (!V[i])
                dfs(i, ++compn);

        if (t++) cout << endl;
        int a, b;
        while(cin >> a >> b, a|b) {
            cout << (V[a] == V[b]?"CONNECTED":"NOT CONNECTED") << endl;
        }
    }
    return 0;
}
```

uva/298.cpp

```
//298
//Race Tracks
//Graphs;Shortest Path;BFS
#include <iostream>
#include <cstring>
#include <queue>
#define MAX 30
using namespace std;

bool V[MAX][MAX][7][7];
int X, Y;

struct Step {
    int x, y, a, b, v;
    Step() {}
    Step(int x, int y, int a, int b, int v) : x(x), y(y), a(a), b(b), v(v) {}

    bool valid() {
        return x>=0 && x<X && y>=0 && y<Y && a >= -3 && a <= 3 && b >= -3 && b <= 3 && !V[x][y][a+3][b+3];
    }

    void mark() {
        V[x][y][a+3][b+3] = true;
    }

    Step go(int mx, int my) {
        return Step(x+a+mx, y+b+my, a+mx, b+my, v+1);
    }
};

int main() {
    int t; cin >> t; t=0;
    while(cin >> X >> Y) {
        memset(V, 0, sizeof(V));
        int x1, y1, x2, y2;
        cin >> x1 >> y1 >> x2 >> y2;

        int p, px1, px2, py1, py2;
        cin >> p;
        while(p--) {
```

```
            cin >> px1 >> px2 >> py1 >> py2;
            for(int i=px1; i<=px2; i++)
                for(int j=py1; j<=py2; j++)
                    for(int ai=0; ai<=6; ai++)
                        for(int bi=0; bi<=6; bi++)
                            V[i][j][ai][bi] = true;
        }

        bool found = false;
        queue<Step> Q;
        Q.push(Step(x1, y1, 0, 0, 0));

        while(!Q.empty()) {
            Step it = Q.front(); Q.pop();
            if (!it.valid()) continue;
            it.mark();

            if (it.x == x2 && it.y == y2) {
                cout << "Optimal solution takes " << it.v << " hops." << endl;
                found = true;
                break;
            }

            for(int ai=-1; ai<=1; ai++)
                for(int bi=-1; bi<=1; bi++)
                    Q.push(it.go(ai, bi));
        }
        if (!found) cout << "No solution." << endl;
    }
}
```

uva/314.cpp

```
//314
//Robot
//Graphs;Shortest Path;BFS
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#include <climits>
#include <vector>
#define MAX 70
using namespace std;

int G[MAX][MAX], n, m, sx, sy, tx, ty;
bool V[MAX][MAX][4];
string dir;

struct Step {
    int x, y, d, v, p;
    Step() {}
    Step(int x, int y, int d, int v, int p) : x(x), y(y), d(d), v(v), p(p) {}
    Step left(int pp) {
        return Step(x, y, (d+3)%4, v+1, pp);
    }
    Step right(int pp) {
        return Step(x, y, (d+1)%4, v+1, pp);
    }
    bool canGo(int i) {
        return (d==0 && x-i>=1 && !G[x-i][y]) ||
               (d==1 && y+i<m-1 && !G[x][y+i]) ||
               (d==2 && x+i<n-1 && !G[x+i][y]) ||
               (d==3 && y-i>=1 && !G[x][y-i]);
    }
    Step go(int pp, int i) {
        if (d==0) return Step(x-i, y, d, v+1, pp);
        if (d==1) return Step(x, y+i, d, v+1, pp);
        if (d==2) return Step(x+i, y, d, v+1, pp);
        if (d==3) return Step(x, y-i, d, v+1, pp);
    }

};

int main() {
    while(cin >> n >> m, n|m) {
        vector<Step> Q;

        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
```

```
49
50          for(int i=0;i<n;i++)
51              for (int j=0;j<m;j++)
52                  cin >> G[i][j];
53
54          n++; m++;
55          for(int i=n-1;i>=0;i--)
56              for (int j=m-1;j>=0;j--)
57                  if (G[i][j])
58                      G[i+1][j] = G[i][j+1] = G[i+1][j+1] = 1;
59
60          cin >> sx >> sy >> tx >> ty >> dir;
61          if (dir=="north") Q.push_back(Step(sx, sy, 0, 0, -1));
62          if (dir=="east") Q.push_back(Step(sx, sy, 1, 0, -1));
63          if (dir=="south") Q.push_back(Step(sx, sy, 2, 0, -1));
64          if (dir=="west") Q.push_back(Step(sx, sy, 3, 0, -1));
65
66          int ptr = 0;
67          while(ptr < Q.size()) {
68              Step it = Q[ptr];
69              if (it.x == tx && it.y == ty) {
70                  cout << it.v << endl;
71                  break;
72              }
73
74              if (V[it.x][it.y][it.d]) { ptr++; continue; }
75              V[it.x][it.y][it.d] = true;
76
77              Q.push_back(it.left(ptr));
78              Q.push_back(it.right(ptr));
79              for (int i=1; i<=3 && it.canGo(i); i++)
80                  Q.push_back(it.go(ptr, i));
81
82              ptr++;
83          }
84          if (ptr == Q.size()) cout << -1 << endl;
85      }
86  }
```

uva/315.cpp

```
1   //315
2   //Network
3   //Graphs;Finding Articulation Points
4   #include <iostream>
5   #include <cstring>
6   #include <string>
7   #include <sstream>
8   #define MAX 101
9   using namespace std;
10  int G[MAX][MAX], V[MAX], L[MAX], P[MAX], n, gpe;
11
12  void dfs(int u, int v) {
13      V[v] = L[v] = ++gpe;
14      for(int i = 1; i <= n; i++) {
15          if(G[v][i]) {
16              if(!V[i]){
17                  dfs(v, i);
18                  L[v] = min(L[v], L[i]);
19                  if(L[i] >= V[v]) P[v]++;
20              } else if(i != u) {
21                  L[v] = min(L[v], V[i]);
22              }
23          }
24      }
25  }
26
27  int main() {
28      while(cin >> n, n) {
29          memset(G, 0, sizeof(G));
30          memset(V, 0, sizeof(V));
31          memset(L, 0, sizeof(L));
32          memset(P, 0, sizeof(P));
33          gpe = 0;
34
35          int a, b; string s;
36          while(getline(cin, s), s != "0") {
37              stringstream sin(s);
38              sin >> a;
39              while(sin >> b) {
40                  G[a][b] = G[b][a] = 1;
```

```
            }
        }
        dfs(1, 1); P[1]--;
        int cnt = 0;
        for(int i=1; i<=n; i++)
            if (P[i]) cnt++;

        cout << cnt << endl;
    }
}
```

uva/321.cpp

```
//321
//The New Villa
//Graphs;Shortest Path;BFS
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#define MAX 15
using namespace std;



int G[MAX][MAX], C[MAX][MAX], n, m1, m2;
bool V[MAX][1200];
string dir;
struct Step {
    int x, s, v, p;
    int type, room;
    Step() {}
    Step(int x, int s, int v, int p) : x(x), s(s), v(v), p(p) {}
    Step(int x, int s, int v, int p, int type, int room) : type(type), room(room), x(x), s(s), v(v), p(p) {}

    Step change(int pp, int i) {
        return Step(x, s ^ (1<<i), v+1, pp, (s & (1<<i))?2:1, i);
    }
    Step move(int pp, int i) {
        return Step(i, s, v+1, pp, 3, i);
    }
};

vector<Step> Q;

void print(Step step) {
        if (step.p == -1) return;
        print(Q[step.p]);
        if (step.type == 1)
            cout << "- Switch on light in room " << step.room+1 << "." << endl;
        if (step.type == 2)
            cout << "- Switch off light in room " << step.room+1 << "." << endl;
        if (step.type == 3)
            cout << "- Move to room " << step.room+1 << "." << endl;
    }



int main() {
    int tt=0;
    while(cin >> n >> m1 >> m2, n|m1|m2) {
        Q = vector<Step>();

        memset(G, 0, sizeof(G));
        memset(C, 0, sizeof(C));
        memset(V, 0, sizeof(V));

        int a, b;
        for(int i=0;i<m1; i++) {
            cin >> a >> b;
            a--; b--;
            G[a][b] = G[b][a] = 1;
        }
        for(int i=0;i<m2; i++) {
            cin >> a >> b;
            a--;b--;
            C[a][b] = 1;
        }

        Q.push_back(Step(0, 1, 0, -1));

```

```
        int ptr = 0;
        cout << "Villa #" << ++tt << endl;
        while(ptr < Q.size()) {
            Step it = Q[ptr];
            if (it.x == n-1 && it.s == (1<<(n-1))) {
                cout << "The problem can be solved in " << it.v << " steps:" << endl;
                print(it);
                break;
            }

            if (V[it.x][it.s]) { ptr++; continue; }
            V[it.x][it.s] = true;

            for(int i=0; i<n; i++) {
                if (G[it.x][i] && (it.s & (1<<i))) Q.push_back(it.move(ptr, i));
                if (C[it.x][i] && it.x != i) Q.push_back(it.change(ptr, i));
            }

            ptr++;
        }
        if (ptr == Q.size()) cout << "The problem cannot be solved." << endl;
        cout << endl;
    }
}
```

uva/361.cpp

```
//361
//Cops and Robbers
//Math;Geometry;Convex Hull;Monotone Chain
#include <iostream>
#include <cmath>
#include <iomanip>
#include <algorithm>
using namespace std;

struct Point {
    int x, y;

    Point() {}
    Point(int x, int y) : x(x), y(y) {}

    int product(Point a, Point b) {
        return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
    }

    bool left(Point a, Point b) {
        return product(a, b) < 0;
    }

    bool operator <(const Point& p) const {
        if (this->x != p.x) return this->x < p.x;
        return this->y < p.y;
    }

    bool operator ==(const Point& p) const {
        return this->x == p.x and this->y == p.y;
    }

    bool insideSegment(Point a, Point b) {
        return product(a, b) == 0 && min(a,b) < *this && *this < max(a,b);
    }
};

int convexHull(Point* P, int n, Point* S) {
    sort(P, P+n);

    int m=0;
    for(int i=0; i<n; i++) {
        while(m >= 2 && S[m-1].left(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    for(int i=n-1, k=m; i >= 0; i--) {
        while(m >= k+2 && S[m-1].left(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    return m;
```

```
}

bool checkInside(Point* P, int n, Point v) {
    for(int i=0; i<n; i++) {
        int j = (i+1)%n;
        if (v == P[i] || v.insideSegment(P[i], P[j]))
            return true;
    }

    for(int i=0; i<n; i++) {
        int j = (i+1)%n;
        if (!v.left(P[i], P[j]))
            return false;
    }

    return true;
}


Point C[250], R[250], CS[250], RS[250];

int main() {
    int c, r, o, tt=1;
    while(cin >> c >> r >> o, c|r|o) {
        for(int i=0; i<c; i++)
            cin >> C[i].x >> C[i].y;

        for(int i=0; i<r; i++)
            cin >> R[i].x >> R[i].y;

        int cs = convexHull(C, c, CS);
        int rs = convexHull(R, r, RS);

        /*for(int i=0; i<cs; i++) {
            cout << " " << CS[i].x << " " << CS[i].y << endl;
        }*/

        cout << "Data set " << tt++ << ":" << endl;
        for(int i=0; i<o; i++) {
            Point p; cin >> p.x >> p.y;
            cout << "     Citizen at (" << p.x << "," << p.y << ") is ";
            if (checkInside(CS, cs, p) && cs > 2)
                cout << "safe." << endl;
            else if (checkInside(RS, rs, p) && rs > 2)
                cout << "robbed." << endl;
            else
                cout << "neither." << endl;
        }
        cout << endl;

    }
}
```

uva/422.cpp

```
//422
//Word-Search Wonder
//Misc;String Matching;KMP;2D
#include <iostream>
#include <string>
#include <cstring>
#define MAX 105
using namespace std;

char C[MAX][MAX];
int F[MAX];
int n;

void kmp_init(string& P) {
    F[0] = 0;  F[1] = 0;
    int i = 1, j = 0;
    while(i<P.size()) {
        if (P[i] == P[j])
            F[++i] = ++j;
        else if (j == 0)
            F[++i] = 0;
        else
            j = F[j];
    }
}

```

```
bool kmp(string& P, int x, int y, int mx, int my) {
    kmp_init(P);
    int j = 0, m = P.size();

    while(x >= 0 && x < n && y >= 0 && y < n) {
        while(j < m) {
            if (P[j] == C[x][y]) {
                x+=mx; y+=my; j++;
            } else break;
        }
        if (j == m) {
            cout << x-m*mx+1 << "," << y-m*my+1 << " " << x+1-mx << "," << y+1-my << endl;
            return true;
        }
        else if (j == 0) { x+=mx; y+=my; };
        j = F[j];
    }
    return false;
}

int main() {
    cin >> n;
    for(int i=0; i<n; i++)
        for(int j=0; j<n; j++)
            cin >> C[i][j];

    string P;
    while(cin >> P, P!="0") {
        bool result = false;
        for(int i=0; i<n; i++) {
            result = result || kmp(P, i,   0,    0,  1);
            result = result || kmp(P, i,   n-1,  0, -1);
            result = result || kmp(P, 0,   i,    1,  0);
            result = result || kmp(P, n-1, i,   -1,  0);

            result = result || kmp(P, 0,   i,    1,  1);
            result = result || kmp(P, i,   n-1, -1, -1);
            result = result || kmp(P, i,   0,    1,  1);
            result = result || kmp(P, n-1, i,   -1, -1);

            result = result || kmp(P, 0,   i,   -1,  1);
            result = result || kmp(P, i,   0,    1, -1);
            result = result || kmp(P, i,   n-1,  1, -1);
            result = result || kmp(P, n-1, i,   -1,  1);
        }

        if (!result)
            cout << "Not found" << endl;
    }
}
```

uva/424.cpp

```
//424
//Integer Inquiry
//Math;Big Integer
#define MAX 110
#include <iostream>
#include <string>
using namespace std;

int T[MAX];
int n=0;

void add(string& s) {
    int c=0;
    int m = s.size();
    for(int i=0; i<m; i++) {
        int a = s[m-i-1]-'0';
        T[i] += a+c;
        c = T[i]/10;
        T[i] %= 10;
    }
    n = max(n, m);
    while (c) {
        T[m] += c;
        c = T[m]/10;
        T[m] %= 10;
        n = max(n, ++m);
    }

```

```
}

int main() {
    string s;
    while(cin >> s, s!="0")
        add(s);

    for(int i=n-1; i>=0; i--)
        cout << T[i];
    cout << endl;
}
```

uva/458.cpp

```
//458
//The Decoder
//Misc;Ad hoc
#include <iostream>
#include <string>
using namespace std;

int main() {
    for(string s; cin>>s;) {
        for(int i=0; i<s.size(); i++)
            s[i]-=7;
        cout << s << endl;
    }
}
```

uva/473.cpp

```
//473
//Raucous Rockers
//Dynamic Programming;Longest Increasing Subsequence
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 10005
using namespace std;

int S[MAX], T[MAX];
char skip;
int main() {
    int n,t,m,cases;
    cin >> cases;
    while(cases--) {
        cin >> n >> t >> m;
        memset(T, 0x3F, sizeof(T));

        for(int i=1; i<=n; i++) {
            cin >> S[i];
            if (i<n) cin >> skip;
        }

        int k=0;
        T[0] = 0;

        for(int i=1;i<=n;i++) {
            for(int j=k; j>=0; j--) {
                int add = 0;
                if ((T[j]%t)+S[i] > t) add = t-T[j]%t;
                if (T[j]+S[i]+add <= T[j+1]) {
                    T[j+1] = T[j]+S[i]+add;
                    k=max(k, j+1);
                }
            }
        }

        int answer = 0;
        for(int i=0; i<=k && T[i] <= m*t; i++)
            answer = i;

        cout << answer << endl;
        if (cases) cout << endl;
    }

    return 0;
}
```

uva/494.cpp

```cpp
//494
//Kindergarten Counting Game
//Misc;Ad hoc
#include <iostream>
#include <string>
using namespace std;

int main() {
    string s;
    while(getline(cin, s)) {
        bool inside = false;
        int words = 0;
        for(int i=0; i<s.size(); i++) {
            if (s[i] >= 'a' && s[i] <= 'z' || s[i] >= 'A' && s[i] <= 'Z') {
                inside = true;
            } else if (inside) {
                inside = false;
                words++;
            }
        }
        if (inside) words++;
        cout << words << endl;
    }

}
```

uva/573.cpp

```cpp
//573
//The Snail
//Misc;Ad hoc
#include <iostream>
using namespace std;

int main() {
    int h,u,d,f;
    while(cin >> h >> u >> d >> f, h|u|d|f) {
        double current = 0, speed=u;

        for (int i=1;;i++) {
            current += speed;
            speed = max(0.0, speed-f/100.0*u);
            if (current > h) {
                cout << "success on day " << i << endl;
                break;
            }
            current -= d;
            if (current < 0) {
                cout << "failure on day " << i << endl;
                break;
            }
        }
    }

    return 0;
}
```

uva/579.cpp

```cpp
//579
//ClockHands
//Misc;Ad hoc
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

int main() {
    int x, y; char c;
    while(cin >> x >> c >> y, x|y) {
        double a = x*30+(y/2.0);
        double b = y*6.0;
        double r = abs(a-b);
        if (r > 180.0)
            r = 360.0-r;
```

```
17
18          cout << fixed << setprecision(3) << r << endl;
19
20      }
21
22  }
```

uva/591.cpp

```
1   //591
2   //Box of Bricks
3   //Misc;Ad hoc
4   #include <iostream>
5   #include <cstring>
6   #include <cmath>
7   using namespace std;
8
9   int T[100];
10
11  int main() {
12      int n, t=0;
13      while(cin >> n, n) {
14          int s=0;
15          for(int i=0; i<n; i++) {
16              cin >> T[i];
17              s += T[i];
18          }
19          s/=n;
20
21          int r=0;
22          for(int i=0; i<n; i++) {
23              r+=abs(T[i]-s);
24          }
25          cout << "Set #" << ++t << endl;
26          cout << "The minimum number of moves is " << r/2 << "." << endl;
27          cout << endl;
28      }
29  }
```

uva/610.cpp

```
1   //610
2   //Street Directions
3   //Graphs;Finding Bridges
4   #include <iostream>
5   #include <cstring>
6   #include <algorithm>
7   #define MAX 1001
8   using namespace std;
9   int G[MAX][MAX], V[MAX], L[MAX], n, m, gpe;
10
11  void dfs(int u, int v) {
12      V[v] = L[v] = ++gpe;
13      for(int i = 1; i <= n; i++) {
14          if(G[v][i]) {
15              if(!V[i]){
16                  dfs(v, i);
17                  L[v] = min(L[v], L[i]);
18                  if(L[i] <= V[v]) {
19                      G[i][v] = 0;
20                  }
21              } else if(i != u) {
22                  L[v] = min(L[v], V[i]);
23                  G[i][v] = 0;
24              }
25          }
26      }
27  }
28
29  int main() {
30      int tt = 0;
31      while(cin >> n >> m, n|m) {
32          memset(G, 0, sizeof(G));
33          memset(V, 0, sizeof(V));
34          memset(L, 0, sizeof(L));
35          gpe = 0;
36
37          cout << ++tt << endl << endl;
38
```

```
        for(int i=0; i<m; i++) {
            int a, b; cin >> a >> b;
            G[a][b] = G[b][a] = 1;
        }

        for(int i=1; i<=n; i++)
            if (!V[i])
                dfs(i, i);

        for(int i=1; i<=n; i++)
            for(int j=1; j<=n; j++)
                if (G[i][j])
                    cout << i << " " << j <<endl;

        cout << "#" << endl;
    }
}
```

uva/627.cpp

```
//627
//The Net
//Graphs;Shortest Path;BFS
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#define MAX 400
using namespace std;

int G[MAX][MAX], n, m;
bool V[MAX];


struct Step {
    int x, v, p;
    Step() {}
    Step(int x, int v, int p) : x(x), v(v), p(p) {}
};

vector<Step> Q;

void print(Step step, bool first) {
    if (step.p != -1) print(Q[step.p], false);
    cout << step.x << (first?"":" ");
}

int main() {
    while(cin >> n) {
        cout << "-----" << endl;
        memset(G, 0, sizeof(G));

        int a, b;
        for(int i=0; i<n;i++) {
            cin >> a;
            while(cin.get()!='\n') {
                if (cin.peek() == '\n') break;
                cin >> b;
                G[a][b] = true;
            }
        }

        cin >> m;
        for(int i=0;i<m;i++) {
            memset(V, 0, sizeof(V));
            cin >> a >> b;
            Q = vector<Step>();
            Q.push_back(Step(a, 0, -1));

            int ptr = 0;
            while(ptr < Q.size()) {
                Step it = Q[ptr];
                if (it.x == b) {
                    print(it, true);
                    cout << endl;
                    break;
                }

                if (V[it.x]) { ptr++; continue; }
                V[it.x] = true;

```

```
                for(int i=1; i<=n; i++)
                    if (G[it.x][i]) Q.push_back(Step(i,it.v+1,ptr));

                ptr++;
            }
            if (ptr == Q.size()) cout << "connection impossible" << endl;
        }
    }
}
```

uva/652.cpp

```
//652
//Eight
//Graphs;Shortest Path;BFS
#include <iostream>
#include <map>
#include <string>
#include <sstream>
using namespace std;

struct Item {
    int p, x;
    string v;
    char m;

    Item() { }
    Item(int p, string v, int x, char m) : p(p), v(v), x(x), m(m) { }
};

map<string, int> M;
Item Q[400000];
int qq=0;

void add(int p, string v, int i, int j, char m) {
    if (not (i%3 == j%3 ^ i/3 == j/3) || j<0 || j>=9) return;
    swap(v[i], v[j]);
    if (M.find(v) != M.end()) return;
    Q[M[v]=qq++] = Item(p, v, j, m);
}

int main() {
    Q[M["123456780"]=qq++] = Item(-1, "123456780", 8, 'z');

    for(int i=0; i<qq; i++) {
        Item p = Q[i];

        add(i, p.v, p.x, p.x-1, 'r');
        add(i, p.v, p.x, p.x+1, 'l');
        add(i, p.v, p.x, p.x-3, 'd');
        add(i, p.v, p.x, p.x+3, 'u');
    }


    int tt; cin >> tt;
    while(tt--) {
        stringstream ss;
        for(int i=0; i<9; i++) {
            string s; cin >> s;
            if (s=="x") s="0";
            ss << s;
        }

        string s = ss.str();
        if (M.find(s) == M.end()) {
            cout << "unsolvable" << endl;
        } else {
            Item item = Q[M[s]];

            while(item.p != -1) {
                cout << item.m;
                item = Q[item.p];
            }
            cout << endl;
        }
        if (tt) cout << endl;
    }
}
```

uva/719.cpp

```cpp
//719
//Glass Beads
//Misc;String Matching;Suffix Array;Circular
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#define MAX 10050
using namespace std;

int RA[MAX], tempRA[MAX];
int SA[MAX], tempSA[MAX];
int C[MAX];

void suffix_sort(int n, int k) {
    memset(C, 0, sizeof C);

    for (int i = 0; i < n; i++)
        C[RA[(i + k)%n]]++;

    int sum = 0;
    for (int i = 0; i < max(256, n); i++) {
        int t = C[i];
        C[i] = sum;
        sum += t;
    }

    for (int i = 0; i < n; i++)
        tempSA[C[RA[(SA[i] + k)%n]]++] = SA[i];

    memcpy(SA, tempSA, n*sizeof(int));
}

void suffix_array(string &s) {
    int n = s.size();

    for (int i = 0; i < n; i++)
        RA[i] = s[i];

    for (int i = 0; i < n; i++)
        SA[i] = i;

    for (int k = 1; k < n; k *= 2) {
        suffix_sort(n, k);
        suffix_sort(n, 0);

        int r = tempRA[SA[0]] = 0;
        for (int i = 1; i < n; i++) {
            int s1 = SA[i], s2 = SA[i-1];
            bool equal = true;
            equal &= RA[s1] == RA[s2];
            equal &= RA[(s1+k)%n] == RA[(s2+k)%n];

            tempRA[SA[i]] = equal ? r : ++r;
        }

        memcpy(RA, tempRA, n*sizeof(int));
    }
}

int main() {
    int tt; cin >> tt;
    while(tt--) {
        string s; cin >> s;
        suffix_array(s);
        cout << SA[0]+1 << endl;
   }
}
```

uva/760.cpp

```cpp
//760
//DNA Sequencing
//Misc;String Matching;Suffix Array;Longest Common Prefix
#include <iostream>
#include <iomanip>
```

```cpp
#include <cstring>
#include <string>
#include <cmath>
#define MAX 10050
using namespace std;

int RA[MAX], tempRA[MAX];
int SA[MAX], tempSA[MAX];
int C[MAX];
int Phi[MAX], PLCP[MAX], LCP[MAX];

void suffix_sort(int n, int k) {
    memset(C, 0, sizeof C);

    for (int i = 0; i < n; i++)
        C[i + k < n ? RA[i + k] : 0]++;

    int sum = 0;
    for (int i = 0; i < max(256, n); i++) {
        int t = C[i];
        C[i] = sum;
        sum += t;
    }

    for (int i = 0; i < n; i++)
        tempSA[C[SA[i] + k < n ? RA[SA[i] + k] : 0]++] = SA[i];

    memcpy(SA, tempSA, n*sizeof(int));
}

void suffix_array(string &s) {
    int n = s.size();

    for (int i = 0; i < n; i++)
        RA[i] = s[i] - 1;

    for (int i = 0; i < n; i++)
        SA[i] = i;

    for (int k = 1; k < n; k *= 2) {
        suffix_sort(n, k);
        suffix_sort(n, 0);

        int r = tempRA[SA[0]] = 0;
        for (int i = 1; i < n; i++) {
            int s1 = SA[i], s2 = SA[i-1];
            bool equal = true;
            equal &= RA[s1] == RA[s2];
            equal &= RA[s1+k] == RA[s2+k];

            tempRA[SA[i]] = equal ? r : ++r;
        }

        memcpy(RA, tempRA, n*sizeof(int));
    }
}

void lcp(string &s) {
    int n = s.size();

    Phi[SA[0]] = -1;
    for (int i = 1; i < n; i++)
        Phi[SA[i]] = SA[i-1];

    int L = 0;
    for (int i = 0; i < n; i++) {
        if (Phi[i] == -1) {
            PLCP[i] = 0;
            continue;
        }
        while (s[i + L] == s[Phi[i] + L])
            L++;

        PLCP[i] = L;
        L = max(L-1, 0);
    }

    for (int i = 1; i < n; i++)
        LCP[i] = PLCP[SA[i]];
}

int main() {
```

```
     string a, b, s;
     int tt = 0;
     while(getline(cin, a) && getline(cin, b)) {
         if (tt++) cout << endl;
         getline(cin, s);
         s = a+"\1"+b+"\2";
         suffix_array(s);
         lcp(s);

         int maxx = 0;
         for(int i=1; i<s.size(); i++) {
             bool left = SA[i-1]+LCP[i] <= a.size();
             bool right = SA[i]+LCP[i] <= a.size();

             if (LCP[i] && (left^right)) {
                 maxx = max(maxx, LCP[i]);
             }
         }

         if (maxx == 0) {
             cout << "No common sequence." << endl;
             continue;
         }

         string last = "some invalid string";
         for(int i=1; i<s.size(); i++) {
             bool left = SA[i-1]+LCP[i] <= a.size();
             bool right = SA[i]+LCP[i] <= a.size();
             string sub = s.substr(SA[i], maxx);

             if (LCP[i]==maxx && (left^right) && last != sub) {
                 cout << sub << endl;
                 last = sub;
             }
         }
     }
 }
```

uva/796.cpp

```
//796
//Critical Links
//Graphs;Finding Bridges
#include <iostream>
#include <cstring>
#include <string>
#include <sstream>
#include <vector>
#include <algorithm>
#define MAX 101
using namespace std;
int G[MAX][MAX], V[MAX], L[MAX], n, gpe;

struct Ponte {
    int a, b;
    Ponte() { }
    Ponte(int a, int b) : a(min(a, b)), b(max(a, b)) {}
};
bool comp(const Ponte& a, const Ponte& b) { return a.a < b.a || (a.a==b.a && a.b < b.b); }
vector<Ponte> P;

void dfs(int u, int v) {
    V[v] = L[v] = ++gpe;
    for(int i = 0; i < n; i++) {
        if(G[v][i]) {
            if(!V[i]){
                dfs(v, i);
                L[v] = min(L[v], L[i]);
                if(L[i] > V[v]) P.push_back(Ponte(v, i));
            } else if(i != u) {
                L[v] = min(L[v], V[i]);
            }
        }
    }
}

int main() {
    while(cin >> n) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        memset(L, 0, sizeof(L));
```

```
        P.clear();
        gpe = 0;

        int a, an, b; char skip;
        for(int i=0; i<n; i++) {
            cin >> a >> skip >> an >> skip;
            while(an--) {
                cin >> b; G[b][a] = G[a][b] = 1;
            }
        }

        for(int i=0; i<n; i++)
            if (!V[i])
                dfs(i, i);

        cout << P.size() << " critical links" << endl;
        sort(P.begin(), P.end(), comp);

        for(int i=0;i<P.size(); i++) {
            cout << P[i].a << " - " << P[i].b << endl;
        }
        cout << endl;
    }
}
```

uva/811.cpp

```
//811
//The Fortified Forest
//Math;Geometry;Convex Hull;Monotone Chain
#include <iostream>
#include <cmath>
#include <iomanip>
#include <algorithm>
#define EP 1e-6
using namespace std;

struct Tree {
    int x, y, v, l;

    int product(Tree a, Tree b) {
        return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
    }

    bool left(Tree a, Tree b) {
        return product(a, b) < 0;
    }

    double dist(Tree b) {
        return sqrt(pow(x-b.x, 2.0) + pow(y-b.y, 2.0));
    }

    bool operator <(const Tree& p) const {
        if (this->x != p.x) return this->x < p.x;
        return this->y < p.y;
    }

    bool operator ==(const Tree& p) const {
        return this->x == p.x and this->y == p.y;
    }
};

int convexHull(Tree* P, int n, Tree* S) {
    sort(P, P+n);

    int m=0;
    for(int i=0; i<n; i++) {
        while(m >= 2 && S[m-1].left(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    for(int i=n-1, k=m; i >= 0; i--) {
        while(m >= k+2 && S[m-1].left(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    return m;
}

```

```
55
56  Tree P[20], T[20], S[20];
57
58  int main() {
59      int n, tt=0;
60      while(cin >> n, n) {
61          for(int i=0; i<n; i++) {
62              cin >> P[i].x >> P[i].y >> P[i].v >> P[i].l;
63          }
64          int mini = 0, minv = 1<<30, minc = 1<<30;
65          double mine = 0.0;
66
67          for(int i=0; i<(1<<n); i++) {
68              int value = 0, length = 0, count=0, ts=0;
69              for(int j=0; j<n; j++) {
70                  if (i & 1<<j) {
71                      value += P[j].v;
72                      length += P[j].l;
73                      count++;
74                  } else {
75                      T[ts++] = P[j];
76                  }
77              }
78              if (value > minv) continue;
79              int s = convexHull(T, ts, S);
80
81              double perimeter = 0;
82              for(int j=0; j<s; j++)
83                  perimeter += S[j].dist(S[(j+1)%n]);
84
85              if (length > perimeter-EP && (value < minv || value == minv && count < minc)) {
86                  mini = i;
87                  minc = count;
88                  minv = value;
89                  mine = length - perimeter;
90              }
91          }
92
93          if (tt) cout << endl;
94          cout << "Forest " << ++tt << endl;
95          cout << "Cut these trees:";
96          for(int i=0; i<n; i++)
97              if (mini & 1<<i)
98                  cout << " " << i+1;
99          cout << endl;
100         cout << "Extra wood: " << fixed << setprecision(2) << mine << endl;
101
102     }
103 }
```

uva/820.cpp

```
1   //820
2   //Internet Bandwidth
3   //Graphs;Maximum Flow;Ford-Fulkerson
4   #include <iostream>
5   #include <iomanip>
6   #include <cstring>
7   #include <string>
8   #include <climits>
9   #include <cmath>
10  #define MAX 1006
11  using namespace std;
12
13  int G[MAX][MAX], n;
14  int F[MAX][MAX];
15  bool V[MAX];
16
17  int send(int s, int t, int minn) {
18      V[s] = true;
19
20      if (s==t) return minn;
21
22      for(int i=1; i<=n; i++) {
23          int capacity = G[s][i]-F[s][i];
24          if (!V[i] && G[s][i]-F[s][i] > 0) {
25              if (int sent = send(i, t, min(minn, capacity))) {
26                  F[s][i] += sent;
27                  F[i][s] -= sent;
28                  return sent;
29              }
```

```
        }
    }

    return 0;
}

int main() {
    int tt=0;
    while(cin >> n, n) {
        memset(G, 0, sizeof(G));
        memset(F, 0, sizeof(F));
        memset(V, 0, sizeof(V));

        tt++;
        int s, t, c;
        cin >> s >> t >> c;
        for(int i=0;i<c;i++) {
            int a, b, f;
            cin >> a >> b >> f;
            G[a][b] = G[b][a] += f;
        }

        int total = 0;
        while(int sent = send(s, t, INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }

        cout << "Network " << tt << endl;
        cout << "The bandwidth is " << total << "." <<  endl;
        cout << endl;
    }


}
```

uva/825.cpp

```
//825
//Walking on the Safe Side
//Dynamic Programming;Ad hoc
#include <iostream>
#include <string>
#include <sstream>
#include <cstring>
#define MAX 250
using namespace std;

int T[MAX][MAX];
int B[MAX][MAX];

int main() {
    int t; cin >> t;
    for(int tt=1; tt<=t; tt++) {
        int r, c; cin >> r >> c;

        string s;
        getline(cin, s);
        for(int i=0; i<r; i++) {
            getline(cin, s);
            stringstream sin(s);

            int a, b; sin >> a;
            while(sin >> b) {
                B[a-1][b-1] = tt;
            }
        }

        for(int i=0; i<r; i++) {
            for(int j=0; j<c; j++) {
                int s = (i==0 && j==0 ? 1 : 0);
                if (i>0) s+= T[i-1][j];
                if (j>0) s+= T[i][j-1];
                T[i][j] = (B[i][j] != tt ? s : 0);
            }
        }

        if (tt>1) cout << endl;
        cout << T[r-1][c-1] << endl;
    }
```

```
}
```

uva/902.cpp

```
//902
//Password Search
//Misc;STL map
#include <iostream>
#include <string>
#include <map>
using namespace std;

map<string, int> T;

int main() {
    int n; string s;
    while(cin >> n >> s) {
        T.clear();

        int maxx=0, maxv=0;
        for(int i=0; i<=s.size()-n; i++){
            int v = ++T[s.substr(i, n)];
            if (v > maxv) {
                maxv = v;
                maxx = i;
            }
        }
        cout << s.substr(maxx, n) << endl;
    }
}
```

uva/907.cpp

```
//907
//Winterim Backpacking Trip
//Dynamic Programming;Integer partition
#define MAX 602
#include <iostream>
#include <cstring>
#include <climits>
using namespace std;

int T[MAX][MAX], S[MAX], n, k;

int main() {
    while(cin >> n >> k) {
        n++; k++;
        memset(T, 0, sizeof(T));

        S[0] = 0;
        for(int i=1; i<=n; i++) {
            cin >> S[i];
        }

        for(int i=1; i<=n; i++)
            T[i][1] = T[i-1][1]+S[i];

        for(int i=1; i<=k; i++)
            T[1][i] = S[1];

        for(int i=2; i<=n; i++) {
            for(int j=2; j<=k; j++) {
                T[i][j] = INT_MAX;
                for(int x=1; x<i; x++)
                    T[i][j] = min(T[i][j], max(T[x][j-1], T[i][1] - T[x][1]));
            }
        }

        cout << T[n][k] << endl;
    }
}
```

uva/908.cpp

```
//908
//Re-connecting Computer Sites
//Graphs;Minimum Spanning Tree;Prim
```

```cpp
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 200010

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    bool operator < (const Road& that) const { return c > that.c; }
};

using namespace std;

vector<Road> G[MAX];

int main() {
    int n, k, m, t=0;
    while(cin >> n) {
        memset(G, 0, sizeof(G));

        int before = 0;
        for(int i=0; i<n-1; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
            before += c;
        }

        int total=0;
        cin >> k;
        for(int i=0; i<k; i++) {
            int a, b, c;
            cin >> a >> b >> c;

            int maxx = 0, maxv, side, counter;
            for(int j=0; j<G[a].size(); j++)
                if (G[a][j].c > maxx) {
                    maxx = G[a][j].c;
                    maxv = j;
                    side= a; counter = b;
                }

            for(int j=0; j<G[b].size(); j++)
                if (G[b][j].c > maxx) {
                    maxx = G[b][j].c;
                    maxv = j;
                    side= b; counter = a;
                }

            if (maxx <= c) continue;
            total = maxx-c;
            G[side][maxv].v = counter;
            G[side][maxv].c = c;
        }

        cin >> m;
        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
        }

        if (t++) cout << endl;
        cout << before << endl << before-total << endl;
    }
    return 0;
}
```

uva/929.cpp

```cpp
//929
//Number Maze
//Graphs;Shortest Path;Dijkstra
#include <iostream>
#include <cstring>
#include <queue>
#include <algorithm>
#define MAX 1001
```

```
using namespace std;

struct Edge {
    int x, y, c;
    Edge(int x, int y, int c) : x(x), y(y), c(c) {};
    inline bool operator <(const Edge& a) const {
        return c > a.c;
    }
};

int G[MAX][MAX], V[MAX][MAX];
int n, m;
priority_queue<Edge> Q;

void try_q(int x, int y, int c) {
    if (x <= 0 || x > n || y<=0 || y > m || c+G[x][y] >= V[x][y]) return;
    Q.push(Edge(x, y, c+G[x][y]));
}


int main() {
    int T; cin >> T;
    for(int tt=1; tt<=T; tt++) {
        memset(V, 0x7f, sizeof(V));
        Q = priority_queue<Edge>();

        cin >> n >> m;
        for(int i=1; i<=n; i++)
            for(int j=1; j<=m; j++)
                cin >> G[i][j];

        Q.push(Edge(1, 1, G[1][1]));

        while(!Q.empty()) {
            Edge e = Q.top(); Q.pop();
            if (V[e.x][e.y] <= e.c) continue;

            V[e.x][e.y] = e.c;

            if (e.x == n && e.y == m) break;

            try_q(e.x-1, e.y, e.c);
            try_q(e.x+1, e.y, e.c);
            try_q(e.x, e.y-1, e.c);
            try_q(e.x, e.y+1, e.c);
        }

        cout << V[n][m] << endl;
    }
}
```

uva/986.cpp

```
//986
//How Many?
//Dynamic Programming;Ad hoc
#include <iostream>
#include <cstring>
using namespace std;

int T[50][50][50][2];

int main() {
    int n, r, h;

    while(cin >> n >> r >> h) {
        memset(T, 0, sizeof(T));
        T[0][0][0][0] = 1;

        for(int i=1; i<=2*n; i++) {
            for(int j=0; j<=2*n; j++) {
                for(int k=0; k<=2*n; k++) {
                    if (j<2*n) {
                        T[i][j][k][0] += T[i-1][j+1][k][0];

                        if (j+1==h && k>0)
                            T[i][j][k][0] += T[i-1][j+1][k-1][1];
                        else if (j+1!=h)
                            T[i][j][k][0] += T[i-1][j+1][k][1];
                    }

```

```
                    if (j>0) {
                        T[i][j][k][1] += T[i-1][j-1][k][0] + T[i-1][j-1][k][1];
                    }

                }
            }
        }

        cout << T[n*2][0][r][0] << endl;

    }
}
```

uva/1016.cpp

```
//1016
//Silly Sort
//Misc;Permutation Cycle
#include <iostream>
#include <algorithm>
#include <cstring>
#define MAX 2000
using namespace std;

int T[MAX], Q[MAX], M[MAX];
bool V[MAX];

int main() {
    int n, t=0;
    while(cin >> n, n) {
        memset(V, 0, sizeof(V));

        int minn=1<<30;
        for(int i=0; i<n; i++) {
            cin >> T[i];
            minn = min(minn, T[i]);
        }

        memcpy(Q, T, sizeof(T));
        sort(Q, Q+n);
        for(int i=0; i<n; i++) {
            M[Q[i]] = i;
        }

        int answer = 0;
        for(int i=0; i<n; i++) {
            if (V[i]) continue;

            int cycle = 0, minc=1<<30, sumc=0;
            for(int j=i; !V[j]; j=M[T[j]]) {
                V[j] = true;
                sumc += T[j];
                minc = min(minc, T[j]);
                cycle++;
            }

            answer += min(sumc + (cycle-2)*minc, sumc + minc + minn * (cycle+1));
        }

        cout << "Case " << ++t << ": " << answer << endl << endl;
    }
}
```

uva/1056.cpp

```
//1056
//Degrees of Separation
//Graphs;Shortest Path;Floyd-Warshall
#include <iostream>
#include <cstring>
#include <string>
#include <map>
#include <cassert>
#define MAX 51

using namespace std;

int G[MAX][MAX];
int n, m;
```

```
map<string, int> M;

int person(string& s) {
    if (M.find(s) != M.end())
        return M[s];
    else
        return M[s]=M.size();

}

int main() {
    int t=0;
    while(cin >> n >> m, n|m) {
        memset(G, 0x1f, sizeof(G));
        M.clear();

        for(int i=0; i<m; i++) {
            string p, q;
            cin >> p >> q;
            int a = person(p), b=person(q);
            G[a][b] = G[b][a] = 1;
        }
        for(int i=1; i<=n; i++) G[i][i] = 0;

        assert(M.size() <= n);

        for(int k=1; k<=n; k++)
            for(int i=1; i<=n; i++)
                for(int j=1; j<=n; j++)
                    G[i][j] = min(G[i][j], G[i][k] + G[k][j]);

        int maxx = 0;
        for(int i=1; i<=n; i++)
                for(int j=1; j<=n; j++)
                    maxx = max(maxx, G[i][j]);

        cout << "Network " << ++t << ": ";

        if (maxx <= n)
            cout << maxx << endl;
        else
            cout << "DISCONNECTED" << endl;

        cout << endl;


    }
    return 0;
}
```

uva/1158.cpp

```
//1158
//CubesSquared
//Dynamic Programming;Knapsack;Infinite Items Knapsack
#include <iostream>
#include <cstring>
#include <vector>
using namespace std;

int K[400001];
vector<int> W;

int main() {

    for(int i=1; i*i*i<=400000; i++)
        W.push_back(i*i*i);

    for(int a=1, i=1; a<=400000; i++, a+=i*i)
        W.push_back(a);

    memset(K, 0x3f, sizeof(K));
    K[0] = 0;
    for(int i=0; i<W.size(); i++)
        for(int j=W[i]; j<=400000; j++)
            K[j] = min(K[j], K[j-W[i]]+1);

    int n;
    while(cin >> n, n!=-1)
        cout << K[n] << endl;

```

```
    return 0;
}
```

uva/1174.cpp

```
//1174
//IP-TV
//Graphs;Minimum Spanning Tree;Prim;Priority Queue
#include <iostream>
#include <cstring>
#include <climits>
#include <string>
#include <vector>
#include <algorithm>
#include <queue>
#include <map>
#define MAX 200010

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};

vector<Road> G[MAX];
priority_queue<Road> Q;
int n, m;
bool V[MAX];
map<string, int> M;

int city(string& s) {
    if (M.find(s) != M.end())
        return M[s];
    else
        return M[s]=M.size();

}

int main() {
    int t; cin >> t; t=0;

    while(cin >> n >> m) {
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        M.clear();
        Q = priority_queue<Road>();

        for(int i=0; i<m; i++) {
            string p, q; int a, b, c;
            cin >> p >> q >> c;
            a = city(p); b=city(q);
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
        }

        int total = 0, totalc=0;

        Q.push(Road(1, 0));

        while(totalc < n) {
            Road item = Q.top(); Q.pop();
            if (V[item.v]) continue;

            V[item.v] = true;
            total += item.c;
            totalc++;

            for(int j=0; j<G[item.v].size(); j++)
                if (!V[G[item.v][j].v])
                    Q.push(G[item.v][j]);
        }

        if (t++) cout << endl;
        cout << total << endl;
    }
    return 0;
}
```

uva/1197.cpp

```cpp
//1197
//The Suspects
//Graphs;DFS
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
using namespace std;

vector<int> G[501], P[30001];
bool VG[501], VP[30001];
int n, m;

int dfs(int v) {
    int sum = 1;
    VP[v] = true;

    for(int i=0; i<P[v].size(); i++) {
        int g = P[v][i];
        if (VG[g]) continue;
        VG[g] = true;

        for(int j=0; j<G[g].size(); j++) {
            int u = G[g][j];
            if (VP[u]) continue;
            sum += dfs(u);
        }
    }

    return sum;
}

int main() {
    while(cin >> n >> m, n|m) {
        memset(G, 0, sizeof(G));
        memset(P, 0, sizeof(P));
        memset(VG, 0, sizeof(VG));
        memset(VP, 0, sizeof(VP));

        for(int i=0; i<m; i++) {
            int k; cin >> k;
            while(k--) {
                int a; cin >> a;
                G[i].push_back(a);
                P[a].push_back(i);
            }
        }

        cout << dfs(0) << endl;
    }
    return 0;
}
```

uva/1200.cpp

```cpp
//1200
//A DP Problem
//Misc;String parsing
#include <iostream>
#include <string>
#include <cmath>
using namespace std;

int getSign(string& s, int &i) {
    if (s[i] == '+') { i++; return 1; }
    if (s[i] == '-') { i++; return -1; }
    return 1;
}

int getNumber(string& s, int& i, bool& got) {
    int result = 0;
    while(s[i] >='0' && s[i] <= '9') {
        result = result*10 + (s[i]-'0');
        i++;
        got = true;
    }
    return result;
```

```
}

bool getX(string& s, int& i) {
    return i<s.size() && s[i] == 'x' && ++i;
}

bool willChange(string& s, int& i) {
    return i<s.size() && s[i] == '=' && ++i;
}


int main()
{
    int t;
    string s;
    cin >> t;

    while(cin >> s) {
        int i=0, A=0, B=0, masterSign = 1;
        while(i<s.size()) {
            int sign = getSign(s, i);
            bool got = false;
            int number = getNumber(s, i, got);
            bool isX = getX(s, i);
            if (isX && !got) number = 1;
           // cout << masterSign << " " << sign << " " << number << " " << isX << endl;

            if (isX)
                B += -1*masterSign*sign*number;
            else
                A += masterSign*sign*number;
            if (willChange(s, i)) masterSign *= -1;
        }
        if (A==0 && B==0) {
            cout << "IDENTITY" << endl;
        } else if (B==0) {
            cout << "IMPOSSIBLE" << endl;
        } else {
            cout << (int)floor(((double)A/B))<< endl;
        }

    }

    return 0;
}
```

uva/1203.cpp

```
//1203
//Argus
//Misc;Priority queue
#include<cstdio>
#include<iostream>
#include<queue>
#include<string>
using namespace std;

#define SZ 3200

struct Item{
  int p, q, b;

  Item() {}
  Item(int q, int p) : p(p), q(q), b(p) {}
  Item(int q, int p, int b) : p(p), q(q), b(b) {}

  inline bool operator < (const Item &d) const{
    if(this->p==d.p) return d.q<this->q;
    return this->p>d.p;
  }

  Item next() {
    return Item(q, p+b, b);
  }

};

priority_queue<Item> Q;
int q, p;

int main(void) {
```

```
  string s;
  int q, p, k;

  while(cin >> s, s!="#") {
    cin >> q >> p;
    Q.push(Item(q, p));
  }

  cin >> k;
  for(int i=0; i<k; ++i) {
    Item item = Q.top(); Q.pop();
    cout << item.q << endl;
    Q.push(item.next());
  }

  return 0;
}
```

uva/1205.cpp

```
//1205
//Color a Tree
//Graphs;Job Scheduling
#include <iostream>
#include <vector>
#include <set>
#define MAX 1008
using namespace std;

struct Cost {
    int a, b, t, v;
    Cost() { }
    Cost(int a, int b, int t, int v) : a(a), b(b), t(t), v(v) {}

    inline bool operator <(const Cost &c) const {
        int cra = a*c.t, crb = c.a*t;
        if (cra!=crb) return cra>crb;
        return v<c.v;
    }
};

int P[MAX], M[MAX];
set<Cost> S;
Cost C[MAX];

int findParent(int v) {
    if (M[v] == v) return v;
    return M[v] = findParent(M[v]);
}

int main() {
    int n, r;
    while(cin >> n >> r, n|r) {
        for(int i=1; i<=n; i++) {
            int a; cin >> a;

            P[i] = 0;
            M[i] = i;
            C[i] = *S.insert(Cost(a, a, 1, i)).first;
        }

        for(int i=1; i<=n-1; i++) {
            int u, v; cin >> u >> v;
            P[v] = u;
        }

        int total = 0, time = 0;
        while(!S.empty()) {
            Cost c = *S.begin();
            int pid = findParent(P[c.v]);
            if (pid == 0) {
                total += time * c.a + c.b;
                time += c.t;
                S.erase(c);
                M[c.v] = 0;
            } else {
                Cost d = *S.find(C[pid]);
                Cost e(c.a + d.a, c.b + d.b + c.a * d.t, c.t + d.t, d.v);

                S.erase(c);
                S.erase(d);
```

```
                S.insert(e);

                M[c.v] = d.v;
                C[e.v] = e;
            }
        }
        cout << total << endl;

    }
}
```

uva/1207.cpp

```
//1207
//AGTC
//Dynamic Programming;Edit Distance
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
string P, Q;


int main() {
    int p, q;
    while(cin >> p >> P >> q >> Q) {
        for(int i=0; i<=p; i++) { T[i][0] = i; }
        for(int i=0; i<=q; i++) { T[0][i] = i; }

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                if (P[i-1] == Q[j-1])
                    T[i][j] = T[i-1][j-1];
                else
                    T[i][j] = min(min(T[i-1][j], T[i][j-1]), T[i-1][j-1])+1;
            }
        }

        cout << T[p][q] << endl;
    }

    return 0;
}
```

uva/1208.cpp

```
//1208
//Oreon
//Graphs;Minimum Spanning Tree;Prim
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#define MAX 501

using namespace std;

int G[MAX][MAX], n;
bool V[MAX];
int D[MAX], DO[MAX];

struct Item {
    int p, a, b;
    Item(){}
    Item(int p, int a, int b) : p(p), a(min(a,b)), b(max(a,b)) {}
};

bool comp(const Item& a, const Item& b) {
    if (a.p != b.p) return a.p < b.p;
    if (a.a != b.a) return a.a < b.a;
    if (a.b != b.b) return a.b < b.b;
    return false;
}

```

```
vector<Item> R;

int updateD(int i) {
    D[i] = 0;
    for(int j=0; j<n; j++) {
        if (G[i][j] && G[i][j] < D[j]) {
            D[j] = G[i][j];
            DO[j] = i;
        }
    }
}

int main() {
    int t; char skip;
    cin >> t;
    t = 0;
    while(cin >> n) {
        memset(V, 0, sizeof(V));
        memset(D, 0x3F, sizeof(D));
        R.clear();

        for(int i=0; i<n; i++) {
            for(int j=0; j<n; j++) {
                cin >> G[i][j];
                if (j+1<n) cin >> skip;
            }
        }

        int total = 0;

        V[0] = true;
        updateD(0);

        for(int k=1; k<n; k++) {
            int minn=INT_MAX, minv;
            for(int i=0; i<n; i++) {
                if (!V[i] && D[i] < minn) {
                    minn = D[i];
                    minv = i;
                }
            }
            R.push_back(Item(minn, DO[minv], minv));
            V[minv] = true;
            updateD(minv);
            total += minn;
        }

        sort(R.begin(), R.end(), comp);
        cout << "Case " << ++t << ":" << endl;
        for(int i=0; i<R.size(); i++) {
            cout << (char)(R[i].a+'A') << "-" << (char)(R[i].b+'A') << " " << R[i].p << endl;
        }
    }
    return 0;
}
```

uva/1213.cpp

```
//1213
//Sum of Different Primes
//Dynamic Programming;Knapsack;Counting Knapsack
#include <iostream>
#include <vector>
#include <cstring>
using namespace std;

long K[20][1300];
bool P[1300];
vector<int> W();

int main() {
    int n, k;

    memset(P, true, sizeof(P));
    P[0] = P[1] = false;
    for(int i=2; i<1300; i++) {
        if (P[i]) {
            W.push_back(i);
            for(int j=i*i; j<1300; j+=i)
                P[j] = false;
        }
```

```
    }

    K[0][0] = 1;
    for(int i=0; i<W.size(); i++)
        for(int p=19; p>0; p--)
            for(int j = W[i]; j<1300; j++)
                K[p][j]+=K[p-1][j-W[i]];

    while(cin >> n >> k, n|k)
        cout << K[k][n] << endl;
}
```

uva/1215.cpp

```
//1215
//String Cutting
//Misc;Binary Search
#include <iostream>
#include <string>
#include <set>
using namespace std;

int T[10001][26];
int C[1001];
set<int> K;

int main() {
    int t; cin >> t; t=0;
    int k; string s;
    while(cin >> k) {
        K.clear();
        for(int i=0; i<k; i++)
            cin >> C[i];

        cin >> s;
        for(int i=1; i<=s.size(); i++)
            for(int j=0; j<26; j++)
                T[i][j] = T[i-1][j] + (s[i-1] == j+'a');


        K.insert(0);
        K.insert(s.size());

        int total = 0;
        for(int i=0; i<k; i++) {
            int mid = C[i];
            set<int>::iterator it = K.lower_bound(mid);
            int hi = *it; it--;
            int lo = *it;

            for(int j=0; j<26; j++) {
                int sidea = T[mid][j]-T[lo][j];
                int sideb = T[hi][j]-T[mid][j];

                if (sidea>0 ^ sideb>0) total++;
            }
            K.insert(mid);
        }
        cout << total << endl;
    }
}
```

uva/1216.cpp

```
//1216
//The Bug Sensor Problem
//Graphs;DFS
#include <iostream>
#include <cstring>
#include <cmath>
#define MAX 1000
using namespace std;

double G[MAX][MAX];
int X[MAX], Y[MAX], n, k;
int V[MAX];

void dfs(int v, int comp, int max) {
    V[v] = comp;
```

```
    for(int i=0; i<n; i++) {
        if (!V[i] && G[v][i] <= max)
            dfs(i, comp, max);
    }
}

int main() {
    int t; cin >> t; t=0;
    while(cin >> k) {
        n=0;

        double maxd=0;
        while(cin >> X[n], X[n]!=-1) {
            cin >> Y[n];
            for(int i=0; i<n; i++) {
                G[i][n] = G[n][i] = sqrt(pow(X[n]-X[i], 2.0)+pow(Y[n]-Y[i], 2.0));
                maxd = max(maxd, G[i][n]);
            }
            n++;
        }

        int begin=0, end=(int)ceil(maxd);
        int best, last = -1;
        while(begin <= end) {
            int mid = (begin+end)/2;
            if (mid == last) break;

            int comp=0;
            memset(V, 0, sizeof(V));
            for(int i=0; i<n; i++)
                if (!V[i])
                    dfs(i, ++comp, mid);

            last = mid;
            if (comp > k)
                begin = mid;
            else {
                if (comp == k) best = mid;
                end = mid;
            }
        }

        cout << best << endl;
    }
}
```

uva/1220.cpp

```
//1220
//Party at Hali-Bula
//Graphs;DFS
#include <iostream>
#include <string>
#include <map>
#include <cstring>
#include <vector>
#define MAX 205
using namespace std;

map<string, int> E;
int emp(string& s) {
    if (E.find(s) != E.end())
        return E[s];
    else
        return E[s] = E.size()-1;
}

bool L[MAX], L2[MAX];
vector<int> G[MAX];
int n;

int dfs(int v) {
    int acum = 0, illu = 0;
    for(int i=0;i<G[v].size();i++) {
        acum += dfs(G[v][i]);
        if (L[G[v][i]]) illu++;
    }
    if (G[v].size() > 0 && illu < G[v].size())
        L[v] = true;
    return acum + L[v];
}
```

```
34
35  int main()
36  {
37      while(cin >> n, n) {
38          memset(G, 0, sizeof(G));
39          memset(L, 0, sizeof(L));
40          E.clear();
41          string a, b;
42          cin >> a;
43          emp(a);
44          for(int i=1;i<n;i++) {
45              cin >> a >> b;
46              G[emp(b)].push_back(emp(a));
47          }
48
49          int total = dfs(0);
50
51          memcpy(L2, L, sizeof(L));
52          bool unique = true;
53          for(int i=0; i<n; i++) {
54              if (!L2[i]) {
55                  memset(L, 0, sizeof(L));
56                  L[i] = true;
57                  if (dfs(0) == total) {
58                      unique = false;
59                      break;
60                  }
61              }
62          }
63
64          cout << n-total << " " << (unique?"Yes":"No") << endl;
65      }
66      return 0;
67  }
```

uva/1223.cpp

```
1   //1223
2   //Editor
3   //Dynamic Programming;Longest Common Substring
4   #include <iostream>
5   #include <string>
6   #include <cstring>
7   #define MAX 5001
8   using namespace std;
9
10  int T[MAX][MAX];
11
12  int main() {
13      int t; cin >> t; t=0;
14      string s;
15      while(cin >> s) {
16          int sz = s.size();
17          int maxx = 0;
18          for(int i=1; i<=sz; i++) {
19              for(int j=1; j<=sz; j++) {
20                  if (s[i-1] == s[j-1] && i!=j)
21                      maxx = max(maxx, T[i][j] = T[i-1][j-1]+1);
22                  else
23                      T[i][j] = 0;
24              }
25          }
26
27          cout << maxx << endl;
28      }
29
30      return 0;
31  }
```

uva/1229.cpp

```
1   //1229
2   //Sub-Dictionary
3   //Graphs;Strongly Connected Components
4   #include <iostream>
5   #include <map>
6   #include <string>
7   #include <cstring>
8   #include <sstream>
```

```
 9  #include <set>
10  #include <algorithm>
11  #define MAX 101
12  using namespace std;
13
14  map<string, int> P;
15  int word(const string& p) {
16      if (P.find(p) != P.end())
17          return P[p];
18      else
19          return P[p] = P.size();
20  }
21
22  int O[MAX], npv, CO[MAX], GR[MAX];
23  string W[MAX];
24  bool G[MAX][MAX], V[MAX];
25  int n;
26  set<int> words;
27  set<string> answer;
28
29  void DFS(int v){
30      V[v] = true;
31      for(int i = 1; i <= n; i++)
32          if (G[v][i] && !V[i])
33              DFS(i);
34      O[npv++] = v;
35  }
36
37  int DFSt(int v, int comp){
38      int acum = 1;
39      V[v] = true; CO[v] = comp;
40      for(int i = 1; i <= n; i++)
41         if (G[i][v] && !V[i])
42              acum += DFSt(i, comp);
43      return acum;
44  }
45
46  void DFSf(int v){
47      V[v] = true;
48      answer.insert(W[v]);
49      for(int i = 1; i <= n; i++)
50          if (G[v][i] && !V[i])
51              DFSf(i);
52  }
53
54  int main() {
55      string s, p, q;
56      while(cin >> n, n) {
57          memset(G, 0, sizeof(G));
58          memset(CO, 0, sizeof(CO));
59          memset(GR, 0, sizeof(GR));
60          P.clear();
61          words.clear();
62          answer.clear();
63          getline(cin, p);
64
65          for(int i=0;i<n; i++) {
66              getline(cin, s);
67              stringstream sin(s);
68              sin >> p;
69              while(sin >> q) {
70                  G[word(p)][word(q)] = true;
71                  GR[word(p)]++;
72              }
73              W[word(p)] = p;
74          }
75
76          npv = 1;
77          memset(V, 0, sizeof(V));
78          memset(O, 0, sizeof(O));
79
80          for(int i = 1; i <= n; i++)
81              if(!V[i]) DFS(i);
82
83          memset(V, 0, sizeof(V));
84
85          int comp = 0;
86          for(int i = n; i > 0; i--) {
87              if(!V[O[i]]) {
88                  comp++;
89                  if (DFSt(O[i], comp) > 1 || GR[O[i]] == 0) {
90                      for(int j=1;j<=n;j++) {
```

```
                        if (CO[j] == comp) {
                            words.insert(j);
                        }
                    }
                }
            }
        }

        memset(V, 0, sizeof(V));
        for(set<int>::iterator it=words.begin(); it!=words.end(); it++) {
            DFSf(*it);
        }

        cout << answer.size() << endl;
        for(set<string>::iterator it=answer.begin(); it!=answer.end(); it++)
            cout << (it!=answer.begin()?" ":"") << *it;
        cout << endl;
    }

    return 0;
}
```

uva/1231.cpp

```
//1231
//ACORN
//Dynamic Programming;Ad hoc
#include <iostream>
#include <string>
#include <cstring>
#define MAX 2001
using namespace std;

int S[MAX][MAX], T[MAX][MAX], M[MAX];
char skip;
int main() {
    int cases; cin >> cases;
    while(cases--) {
        memset(S, 0, sizeof(S));
        memset(M, 0, sizeof(M));
        int t, h, f;
        cin >> t >> h >> f;

        for(int i=0; i<t; i++) {
            int k, a; cin >> k;
            while(k--) {
                cin >> a;
                S[a][i]++;
            }
        }

        for(int i=h; i>=0; i--) {
            for(int j=0; j<t; j++) {
                int move = i+f<=h ? M[i+f] : 0;
                int stay = i+1<=h ? T[i+1][j] : 0;
                T[i][j] = max(move, stay) + S[i][j];
                M[i] = max(M[i], T[i][j]);
            }
        }

        cout << M[0] << endl;
    }

    return 0;
}
```

uva/1232.cpp

```
//1232
//SKYLINE
//Misc;Segment Tree
#include <iostream>
#include <string>
#include <set>
using namespace std;

struct Node {
    int a, b, h;
    bool leaf;
```

```
    Node() {}
    Node(int a, int b, int h, bool leaf=true) : a(a), b(b), h(h), leaf(leaf) {}
};

Node H[5000005];
inline int left(int i) { return 2*i; }
inline int right(int i) { return 2*i+1; }

inline void cut(int v, int x) {
    H[left(v)] = Node(H[v].a, x, H[v].h);
    H[right(v)] = Node(x, H[v].b, H[v].h);
    H[v].leaf = false;
}

int dfs(int v, int a, int b, int h) {
    a = max(a, H[v].a);
    b = min(b, H[v].b);
    if (b<=a) return 0;

    if (!H[v].leaf)
        return dfs(left(v), a, b, h) + dfs(right(v), a, b, h);

    if (H[v].h > h) return 0;
    if (H[v].a < a) return cut(v, a), dfs(v, a, b, h);
    if (b < H[v].b) return cut(v, b), dfs(v, a, b, h);

    H[v].h = h;
    return b-a;
}

int main() {
    int n, t; cin >> t; t=0;
    while(cin >> n, n) {
        H[1] = Node(0, 100000, 0);

        int sum = 0;
        while(n--) {
            int a, b, h;
            cin >> a >> b >> h;
            sum += dfs(1, a, b, h);
        }
        cout << sum << endl;

    }
}
```

uva/1233.cpp

```
//1233
//USHER
//Graphs;Shortest Path;Floyd-Warshall
#include <iostream>
#include <cstring>
#include <string>
#include <map>
#include <cassert>
#define MAX 501

using namespace std;

int P[MAX];
int G[MAX][MAX];
int n, m;

int main() {
    int tt; cin >> tt;
    while(tt--) {
        int b, p, q; cin >> b >> p >> q;
        for(int i=0; i<q; i++)
            cin >> P[i];

        memset(G, 0x1f, sizeof(G));

        G[0][0] = 0;
        for(int i=1; i<=p; i++) {
            int k; cin >> k;
            G[i][i] = 0;
            for(int j=0; j<k; j++) {
                int x, y; cin >> x >> y;
                G[i][y] = min(G[i][y], x);
            }
```

```
        }

        for(int k=0; k<=p; k++)
            for(int i=0; i<=p; i++)
                for(int j=0; j<=p; j++)
                    G[i][j] = min(G[i][j], G[i][k] + G[k][j]);

        int minn = 1<<30;
        for(int i=0; i<q; i++) {
            minn = min(minn, G[P[i]][0]);
        }

        int current = 0;
        int answer = 0;
        while(true) {
            if ((current += minn) >= b) break;
            current--;
            answer++;
        }

        cout << answer << endl;
    }
    return 0;
}
```

uva/1234.cpp

```
//1234
//RACING
//Graphs;Minimum Spanning Tree;Prim;Priority Queue
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 10005

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c < that.c; }
};

vector<Road> G[MAX];
int CStart[MAX], CCount[MAX], nc;
priority_queue<Road> Q;
vector<int> R;
int n, m;
bool V[MAX];

int dfs(int v) {
    V[v] = true;
    int acum = 1;
    for(int i=0; i<G[v].size(); i++)
        if (!V[G[v][i].v])
            acum += dfs(G[v][i].v);
    return acum;
}

int main() {
    int t; cin >> t;
    while(cin >> n >> m, t--) {
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        nc = 0;
        R.clear();

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
        }

        for(int i=1; i<=n; i++) {
            if (!V[i]) {
                CStart[nc]=i;
                CCount[nc]=dfs(i);
```

```
                nc++;
            }
        }

        int result=0;
        for(int i=0; i<nc; i++) {
            int totalc=0;
            Q.push(Road(CStart[i], 0));
            memset(V, 0, sizeof(V));

            while(totalc < CCount[i]) {
                Road item = Q.top(); Q.pop();
                if (V[item.v]) { result+=item.c; continue; }

                V[item.v] = true;
                totalc++;

                for(int j=0; j<G[item.v].size(); j++)
                    if (!V[G[item.v][j].v])
                        Q.push(G[item.v][j]);
            }
            while(!Q.empty()) {
                result += Q.top().c;
                Q.pop();
            }
        }
        cout << result << endl;
    }
    return 0;
}
```

uva/1235.cpp

```
//1235
//Anti Brute Force Lock
//Graphs;Minimum Spanning Tree;Prim
#include <iostream>
#include <cstring>
#include <climits>
#define MAX 501

using namespace std;

int abs(int a) {
    return a>0?a:-a;
}

int d(int a, int b) {
    int result = 0;
    for(int i=0; i<4; i++) {
        int aa=a%10, bb=b%10;
        result += min(abs(aa-bb), 10-abs(aa-bb));
        a/=10; b/=10;
    }
    return result;
}

int K[MAX], G[MAX][MAX], n;
bool V[MAX];
int D[MAX];

int updateD(int i) {
    D[i] = 0;
    for(int j=0; j<n; j++) {
        if (G[i][j]) D[j] = min(D[j], G[i][j]);
    }
}

int main()
{
    int t;
    cin >> t;
    while(cin >> n) {
        memset(V, 0, sizeof(V));
        memset(D, 0x3F, sizeof(D));

        for(int i=0; i<n; i++)
            cin >> K[i];

        for(int i=0; i<n; i++)
            for(int j=i+1; j<n; j++)
```

```
                G[i][j] = G[j][i] = d(K[i],K[j]);

        int total=INT_MAX;
        for(int i=0;i<n;i++) total = min(total, d(0, K[i]));

        V[0] = true;
        updateD(0);

        for(int k=1; k<n; k++) {
            int minn=INT_MAX, minv;
            for(int i=0; i<n; i++) {
                if (!V[i] && D[i] < minn) {
                    minn = D[i];
                    minv = i;
                }
            }
            V[minv] = true;
            updateD(minv);
            total += minn;
        }

        cout << total << endl;
    }
    return 0;
}
```

uva/1239.cpp

```
//1239
//Greatest K-Palindrome Substring
//Dynamic Programming;Ad hoc
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];

int main() {
    int t; cin >> t; t=0;
    string P;
    int k;
    while(cin >> P >> k) {
        int p = P.size();

        int maxx=0;
        for(int i=p; i>=1; i--) {
            for(int j=i; j<=p; j++) {
                T[i][j] = T[i+1][j-1] + (P[i-1] == P[j-1] ? 0 : 1);

                if (T[i][j] <= k)
                    maxx = max(maxx, j-i+1);
            }
        }

        cout << maxx << endl;
    }

    return 0;
}
```

uva/1246.cpp

```
//1246
//Find Terrorists
//Math;Sieve
#include <iostream>
#include <vector>
#include <cstring>
using namespace std;

bool P[100];
int T[10000001];
vector<int> W;

long long real_mod(long long a, long long b) {
    long long c = a%b;
```

```
    if (c<0) c+=b;
    return c;
}

int main() {
    int n, k;

    memset(P, true, sizeof(P));
    P[0] = P[1] = false;
    for(int i=2; i<100; i++) {
        if (P[i]) {
            W.push_back(i);
            for(int j=i*i; j>=0 && j<100; j+=i)
                P[j] = false;
        }
    }

    int t; cin >> t; t=0;
    int a, b;
    while(cin >> a >> b) {
        memset(T, 0, sizeof(int)*(b-a+1));

        if (a==0) { T[0]-=2; T[1] -= 1; }
        if (a==1) { T[0]-=1; }

        for(long long i=2; i*i<=b; i++) {
            for(long long j=max(real_mod(i*i+i-a, i), i*i+i-a); j<=(b-a); j+=i) {
                T[j]+=2;
            }
            int tmp = i*i-a;
            if (tmp >= 0 && tmp <= (b-a))
                T[tmp]++;
        }

        int cnt=0;
        for(int i=0; i<=(b-a);i++) {
            if (P[T[i]+2]) {
                cout << (cnt++?" ":"") << i+a;
            }
        }
        if (!cnt) cout << -1;
        cout << endl;
    }

}
```

uva/1247.cpp

```
//1247
//Interstar Transport
//Graphs;Shortest Path;Dijkstra
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 30

using namespace std;

struct Edge {
    int u, v, c;
    Edge(int u, int v, int c) : u(u), v(v), c(c) {}
    inline bool operator < (const Edge& that) const { return c > that.c; }
};

int G[MAX][MAX];
int V[MAX];
int D[MAX];
int n, m;

void show(int t) {
    if (D[t] != t) {
        show(D[t]);
        cout << " ";
    }

    cout << (char)(t+'A');
}

```

```
int shortest(int a, int b) {
    memset(V, 0x3f, sizeof(V));
    priority_queue<Edge> Q;
    Q.push(Edge(a, a, 0));

    while(!Q.empty()) {
        Edge item = Q.top(); Q.pop();
        if (item.c >= V[item.v]) continue;
        V[item.v] = item.c;
        D[item.v] = item.u;

        for(int j=0; j<n; j++) {
            if (G[item.v][j]) {
                Edge e = Edge(item.v, j, item.c+G[item.v][j]);
                if (e.c <= V[e.v])
                    Q.push(e);
            }
        }
    }
    show(b); cout << endl;
}

int main() {
    while(cin >> n >> m) {
        memset(G, 0, sizeof(G));

        for(int i=0; i<m; i++) {
            char a, b; int c;
            cin >> a >> b >> c;
            G[a-'A'][b-'A'] = G[b-'A'][a-'A'] = c;
        }

        int k; cin >> k;
        while(k--) {
            char a, b; cin >> a >> b;
            shortest(a-'A', b-'A');
        }
    }
    return 0;
}
```

uva/1251.cpp

```
//1251
//Repeated Substitution with Sed
//Graphs;Shortest Path;BFS
#include <iostream>
#include <queue>
#include <string>
#include <set>
#define MAX 1000
using namespace std;

struct Item {
    string s;
    int c;
    Item(string s, int c) : s(s), c(c) {}
};

string replace(string str, string from, string to) {
    if(from.empty())
        return str;
    int start_pos = 0;
    while((start_pos = str.find(from, start_pos)) != string::npos) {
        str.replace(start_pos, from.length(), to);
        start_pos += to.length();
    }
    return str;
}

int n;
string A[MAX], B[MAX];

int main() {
    while(cin >> n, n) {
        for(int i=0; i<n; i++)
            cin >> A[i] >> B[i];
        string a, b;
        cin >> a >> b;

        queue<Item> Q;
```

```
        set<string> S;
        Q.push(Item(a, 0));

        int answer = -1;
        while(!Q.empty()) {
            Item e = Q.front(); Q.pop();
            if (e.s == b) {
                answer = e.c;
                break;
            }

            if (S.find(e.s) != S.end()) continue;
            S.insert(e.s);

            for(int i=0; i<n; i++) {
                string s = replace(e.s, A[i], B[i]);
                if (S.find(s) != S.end() || s.size() > 10) continue;
                Q.push(Item(s, e.c+1));
            }
        }

        cout << answer << endl;

    }

}
```

uva/1263.cpp

```
//1263
//Mines
//Graphs;Topological Sorting
#include <iostream>
#include <cstring>
#define MAX 2001
using namespace std;

int n;
bool V[MAX], G[MAX][MAX];
int X[MAX], Y[MAX], D[MAX], O[MAX], npv;

inline int abs(int a) {
    return a>0?a:-a;
}

void dfs(int v, bool sort){
    V[v] = true;
    for(int i = 1; i <= n; i++)
        if (G[v][i] && !V[i])
            dfs(i, sort);
    if (sort)
        O[++npv] = v;
}

int main() {
    int t; cin >> t; t=0;
    while(cin >> n) {
        memset(G, 0, sizeof(G));
        for(int i=1; i<=n; i++) {
            cin >> X[i] >> Y[i] >> D[i];
        }
        for(int i=1; i<=n; i++) {
            for(int j=1; j<=n; j++) {
                int r = D[i]/2;
                if (abs(X[j]-X[i])<=r && abs(Y[j]-Y[i]) <=r && i!=j)
                    G[i][j] = true;
            }
        }

        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i = 1; i <= n; i++)
            if(!V[i]) dfs(i, true);

        memset(V, 0, sizeof(V));

        int comp = 0;
        for(int i = n; i > 0; i--)
            if(!V[O[i]]) {
```

```
                comp++;
                dfs(O[i], false);
            }

        cout << comp << endl;
    }

    return 0;
}
```

uva/1265.cpp

```
//1265
//Tour Belt
//Graphs;Minimum Spanning Tree;Kruskal
#include <iostream>
#include <cstring>
#include <vector>
#include <set>
#include <algorithm>
#include <cassert>
using namespace std;

struct Edge {
    int x, y, v;
    Edge() {}
    Edge(int x, int y, int v) : x(x), y(y), v(v) {}

    inline bool operator <(const Edge& that) const {
        return this->v > that.v;
    }
};

Edge E[5006*2506];
int A[5006][5006], B[5006][5006];
int P[5002], C[5002];

inline int findset(int v) {
    if (P[v] != v)
        return P[v] = findset(P[v]);
    return v;
}

inline int unionset(int x, int y) {
    int a = findset(x), b = findset(y);
    if (a==b) return 0;
    if (a>b) swap(a,b);
    P[b] = a;
    C[a] += C[b];
    C[b] = 0;
    return a;
}

int main() {
    int tt; cin >> tt;

    while(tt--) {
        int n, m; cin >> n >> m;
        for(int i=1; i<=n; i++) {
            P[i] = i;
            C[i] = 1;

            for(int j=1; j<=n; j++) {
                A[i][j] = 1<<29;
                B[i][j] = 0;
            }
        }

        for(int i=0; i<m; i++) {
            int x, y, v;
            cin >> x >> y >> v;
            E[i] = Edge(x, y, v);

            A[x][y] = A[y][x] = B[x][y] = B[y][x] = v;
        }

        sort(E, E+m);

        int total = 0;
        for(int i=0; i<m; i++) {
            int x = findset(E[i].x), y = findset(E[i].y);
```

```
            if (x==y) continue;

            int a = unionset(x, y);

            int outside = 0, inside = 1<<29;
            for(int j=1; j<=n; j++) {
                A[a][j] = A[j][a] = min(A[x][j], A[y][j]);
                B[a][j] = B[j][a] = max(B[x][j], B[y][j]);

                if (findset(a) == findset(j))
                    inside = min(inside, A[a][j]);
                else
                    outside = max(outside, B[a][j]);
            }

            if (inside > outside)
                total += C[a];
        }

        cout << total << endl;

    }
}
```

uva/10003.cpp

```
//10003
//Cutting Sticks
//Dynamic Programming;Matrix Multiplication
#define MAX 1001
#include <iostream>
#include <cstring>
#include <climits>
using namespace std;

int T[MAX][MAX], S[MAX], n;
bool V[MAX][MAX];

int TT(int a, int b) {
    if (a+1==b) return 0;
    if (V[a][b]) return T[a][b];

    int minn = INT_MAX;
    for(int i=a+1; i<b; i++)
        minn = min(minn, TT(a,i) + TT(i,b) + S[b]-S[a]);

    V[a][b] = true;
    return T[a][b] = minn;
}

int main() {
    int t;
    while(cin >> t, t) {
        cin >> n;

        memset(S, 0, sizeof(S));
        memset(V, 0, sizeof(V));
        S[0] = 0;
        for(int i=1; i<=n; i++) {
            cin >> S[i];
        }
        S[n+1] = t;

        cout << "The minimum cutting is " << TT(0, n+1) << "." << endl;

    }
}
```

uva/10015.cpp

```
//10015
//Joseph's Cousin
//Dynamic Programming;Josephus Problem
#include <iostream>
#include <vector>
#define MAX 35000
#define MAX2 3503
using namespace std;

```

```
bool P[MAX];
vector<int> W;
int T[MAX2][MAX2];

int main() {
    P[0] = P[1] = true;
    for(int i=2; i<MAX; i++) {
        if (!P[i]) {
            W.push_back(i);
            for(int j=i*i; j<MAX; j+=i)
                P[j] = true;
        }
    }

    for(int i=2; i<MAX2; i++)
        for(int j=0; j<MAX2-1; j++)
            T[i][j] = (W[j] + T[i-1][j+1])%i;

    int n;
    while(cin >> n, n) {
        cout << T[n][0]+1 << endl;
    }

}
```

uva/10018.cpp

```
//10018
//Reverse and Add
//Misc;Ad hoc
#include <iostream>
using namespace std;

long reverse(long a) {
    long b = 0;
    while(a) {
        b = b*10 + a%10;
        a /= 10;
    }
    return b;
}

int main() {
    int n;
    cin >> n;
    while(n--) {
        long a; cin >> a;
        for(int i=1; i<=1000; i++ ) {
            a = a + reverse(a);
            if (a == reverse(a)) {
                cout << i << " " << a << endl;
                break;
            }
        }
    }
}
```

uva/10035.cpp

```
//10035
//Primary Arithmetic
//Misc;Ad hoc
#include <iostream>
using namespace std;

int main() {
    int a, b;
    while(cin >> a >> b, a|b) {
        int carries = 0;
        int c = 0;
        while(a|b) {
            int s = a%10 + b%10 + c;
            c = s/10;
            if (c) carries++;

            a/=10;
            b/=10;
        }

```

```
        if (carries == 0) {
            cout << "No carry operation." << endl;
        } else if (carries==1) {
            cout << "1 carry operation." << endl;
        } else {
            cout << carries << " carry operations." << endl;
        }
    }
}
```

uva/10044.cpp

```
//10044
//Erdos Number
//Graphs;Shortest Path;BFS
#include <iostream>
#include <cstring>
#include <climits>
#include <string>
#include <vector>
#include <queue>
#include <map>
#define MAX 5000
using namespace std;

vector<int> G[MAX];
int n, m;
bool V[MAX];
map<string, int> A;

struct Step {
    int x, v;
    Step() {}
    Step(int x, int v) : x(x), v(v) {}
};

queue<Step> Q;

int author(const string& a) {
    if (A.find(a) != A.end())
        return A[a];
    else
        return A[a] = A.size()-1;
}


char C[MAX];
void parseAuthors(const string& s) {
    vector<int> TA;
    int commas = 0, chars=0;
    for(int i=0;i<s.size();i++) {
        char c = s[i];
        if (chars == 0 && c == ' ') continue;

        if ((c==',' || c==':') && ++commas == 2) {
            TA.push_back(author(string(C, chars)));
            chars = commas = 0;
        } else {
            C[chars++] = c;
        }
    }
    for(int i=0;i<TA.size(); i++) {
        for(int j=i+1;j<TA.size(); j++) {
            G[TA[i]].push_back(TA[j]);
            G[TA[j]].push_back(TA[i]);
        }
    }

}

int main() {
    string s;
    int t=0, tt;
    cin >> tt;
    while(t++ < tt) {
        cin >> n >> m;
        memset(G, 0, sizeof(G));
        A.clear();
        getline(cin, s);
        while(n--) {
            getline(cin, s);
```

```
            parseAuthors(s);
        }

        cout << "Scenario " << t << endl;
        for(int i=0;i<m;i++) {
            bool stop;
            memset(V, 0, sizeof(V));
            getline(cin, s);
            int b = author(s);
            Q = queue<Step>();
            Q.push(Step(author("Erdos, P."), 0));
            bool found = false;

            while(!Q.empty()) {
                Step it = Q.front(); Q.pop();
                if (it.x == b) {
                    cout << s << " " << it.v << endl;
                    found = true;
                    break;
                }

                V[it.x] = true;

                for(int i=0; i<G[it.x].size(); i++)
                    if (!V[G[it.x][i]]) Q.push(Step(G[it.x][i], it.v+1));
            }
            if (!found) cout << s << " infinity" << endl;
        }
     }
     return 0;
}
```

uva/10051.cpp

```
//10051
//Tower of Cubes
//Dynamic Programming;Longest Increasing Subsequence
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#include <climits>
#define MAX 501
#define MAXC 101
using namespace std;

int T[MAX][MAXC], F[MAX][MAXC], P[MAX][MAXC];
int A[6];

string translate(int side) {
    switch(side) {
        case 0: return "front";
        case 1: return "back";
        case 2: return "left";
        case 3: return "right";
        case 4: return "top";
        case 5: return "bottom";
    }
}

void print(int first, int k) {
    if (k==0) return;

    print(F[k][first], k-1);
    cout << T[k][first] << " " << translate(P[k][first]) << endl;
}

int main() {
    int n, n2, t;
    while(cin >> n, n) {
        if (t++) cout << endl;
        cout << "Case #" << t << endl;

        memset(T, 0, sizeof(T));

        for(int i=1;i<=MAXC; i++) {
            T[0][i] = 1;
        }
        int k = 0;

        for(int cube=1;cube<=n;cube++) {
```

```
            for(int i=0;i<6;i++) cin >> A[i];
            int newk = k;
            for(int j=k; j>=0; j--) {
                for(int i=0;i<6;i++) {
                    int other = (i/2*2)+(1-i%2);
                    if (T[j][A[i]] && !T[j+1][A[other]]) {
                        T[j+1][A[other]] = cube;
                        F[j+1][A[other]] = A[i];
                        P[j+1][A[other]] = i;
                        newk = max(newk, j+1);
                    }
                }
            }
            k=newk;
        }

        cout << k << endl;

        int first=0;
        for(int i=1;i<=100;i++)
            if (T[k][i]) first=i;

        print(first, k);
    }

    return 0;
}
```

uva/10065.cpp

```
//10065
//Useless Tile Packers
//Math;Geometry;Convex Hull;Monotone Chain
#include <iostream>
#include <cmath>
#include <iomanip>
#include <algorithm>
using namespace std;

struct Point {
    int x, y;

    Point() {}
    Point(int x, int y) : x(x), y(y) {}

    bool left(Point& a, Point& b) {
        return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x) < 0;
    }

    bool operator <(const Point& p) const {
        if (this->x != p.x) return this->x < p.x;
        return this->y < p.y;
    }

    bool operator ==(const Point& p) const {
        return this->x == p.x and this->y == p.y;
    }

};

double area(Point* A, int a) {
    double area = 0;
    for(int i=0; i<a; i++) {
        int j = (i+1)%a;
        area += (A[i].x + A[j].x) * (A[i].y - A[j].y);
    }
    return area / 2;
}

int convexHull(Point* P, int n, Point* S) {
    sort(P, P+n);

    int m=0;
    for(int i=0; i<n; i++) {
        while(m >= 2 && S[m-1].left(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    for(int i=n-1, k=m; i >= 0; i--) {
        while(m >= k+2 && S[m-1].left(S[m-2], P[i])) m--;
```

```
        S[m++] = P[i];
    }
    m--;

    return m;
}

Point P[110], S[110];

int main() {
    int n, tt=0;
    while(cin >> n, n) {
        for(int i=0; i<n; i++) {
            int x, y; cin >> x >> y;
            P[i] = Point(x, y);
        }

        double original = abs(area(P, n));

        int m = convexHull(P, n, S);
        double modified = abs(area(S, m));
        double ratio = 100*(1.0-(original/modified));

        cout << "Tile #" << ++tt << endl;
        cout << "Wasted Space = " << fixed << setprecision(2) << ratio << " %" << endl;
        cout << endl;
    }


}
```

uva/10066.cpp

```
//10066
//The Twin Towers
//Dynamic Programming;Longest Common Subsequence
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 105
using namespace std;

int T[MAX][MAX];
int P[MAX], Q[MAX];

int main() {
    int p, q, tt=0;
    while(cin >> p >> q, tt++, p&&q) {
        memset(T, 0, sizeof(T));

        for(int i=0; i<p;i++) cin >> P[i];
        for(int i=0; i<q;i++) cin >> Q[i];

        for(int i=0; i<=p; i++) T[i][0] = 0;
        for(int i=0; i<=q; i++) T[0][i] = 0;

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                if (P[i-1] == Q[j-1])
                    T[i][j] = T[i-1][j-1] + 1;
                else
                    T[i][j] = max(T[i-1][j], T[i][j-1]);
            }
        }
        cout << "Twin Towers #" << tt << endl;
        cout << "Number of Tiles : " << T[p][q] << endl;
        cout << endl;
    }

    return 0;
}
```

uva/10090.cpp

```
//10090
//Marbles
//Math;Extended Euclid
#include <iostream>
```

```
#define ull long long
using namespace std;

ull euclid(ull a, ull b, ull& rx, ull& ry) {
    if (!b) return rx=1, ry=0, a;

    ull q = a/b;
    ull x, y;
    ull g = euclid(b, a-q*b, x, y);
    return rx=y, ry=x-q*y, g;
}

ull solve(ull a, ull b, ull c) {
    ull x, y;
    ull g = euclid(a, b, x, y);
    if (c%g) return -1;
    ull ag=a/g, bg=b/g, cg=c/g;
    return (x*cg%bg+bg)%bg;
}

int main() {
    ull n, c1, n1, c2, n2;

    while(cin >> n, n) {
        cin >> c1 >> n1 >> c2 >> n2;

        ull sol1=solve(n1, n2, n), sol2=solve(n2, n1, n);

        ull sol12=(n-n1*sol1)/n2, sol21=(n-n2*sol2)/n1;

        if (sol1 < 0 || sol12 < 0) {
            cout << "failed" << endl;
            continue;
        }

        ull cos1=c1*sol1+sol12*c2;
        ull cos2=c2*sol2+sol21*c1;
        if (cos1 < cos2)
            cout << sol1 << " " << sol12 << endl;
        else
            cout << sol21 << " " << sol2 << endl;
    }
}
```

uva/10092.cpp

```
//10092
//The Problem with the Problem Setter
//Graphs;Maximum Flow;Ford-Fulkerson
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#include <climits>
#define MAX 1100
using namespace std;

int G[MAX][MAX], nk, np, n;
bool V[MAX];

int SOURCE() { return 1; }
int P(int i) { return 1+i; }
int K(int i) { return 1+np+i; }
int TARGET() { return 2+np+nk; }

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=1; i<=n; i++) {
        if (!V[i] && G[s][i] > 0) {
            if (int sent = send(i, t, min(minn, G[s][i]))) {
                G[s][i] -= sent;
                G[i][s] += sent;
                return sent;
            }
        }
    }
    return 0;
}
```

```

int main() {
    int tmp, tmp2;
    while(cin >> nk >> np, nk|np) {
        n = nk+np+2;
        int expected = 0;
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));

        for(int i=1; i<=nk; i++) {
            cin >> tmp;
            expected += tmp;
            G[K(i)][TARGET()] = tmp;
        }

        for(int i=1;i<=np; i++) {
            cin >> tmp;
            G[SOURCE()][P(i)] = 1;
            for(int j=0;j<tmp;j++) {
                cin >> tmp2;
                G[P(i)][K(tmp2)] = 1;
            }
        }

        int total = 0;
        while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }
        cout << (expected == total ? 1: 0) << endl;
        if (expected == total) {
            for(int i=K(1); i<=K(nk); i++) {
                bool printed = false;
                for(int j=P(1); j<=P(np); j++) {
                    if (G[i][j]) {
                        cout << (printed?" ":"") << (j-1);
                        printed = true;
                    }
                }
                cout << endl;
            }
        }
    }
}
```

uva/10104.cpp

```
//10104
//Euclid Problem
//Math;Extended Euclid
#include <iostream>
#define ull long long
using namespace std;

int euclid(int a, int b, int& rx, int& ry) {
    if (!b) return rx=1, ry=0, a;

    int q = a/b;
    int x, y;
    int g = euclid(b, a-q*b, x, y);
    return rx=y, ry=x-q*y, g;
}

int main() {
    int a, b;

    while(cin >> a >> b) {
        int x, y;
        int d = euclid(a,b,x,y);

        cout << x << " " << y << " " << d << endl;
    }
}
```

uva/10113.cpp

```
//10113
//Exchange Rates
//Graphs;DFS
```

```cpp
#include <iostream>
#include <vector>
#include <map>
#include <set>
#include <algorithm>
#include <string>
using namespace std;

int gcd(int a, int b) {
    while(b)
        swap(a=a%b,b);
    return a;
}

struct Edge {
    string s;
    int a, b;
    Edge() : a(0), b(0) { }
    Edge(string s, int a, int b) : s(s), a(a), b(b) {}

    Edge next(Edge e) {
        int na = a*e.a, nb = b*e.b;
        int g = gcd(na, nb);
        na /= g; nb /= g;
        return Edge(e.s, na, nb);
    }

    bool valid() { return a!=0; }
};


map<string, vector<Edge> > G;
set<string> V;

Edge dfs(Edge e, string target) {
    if (e.s == target) return e;
    V.insert(e.s);
    vector<Edge> ve = G[e.s];

    for(int i=0; i<ve.size(); i++) {
        if (V.find(ve[i].s) == V.end()) {
            Edge other = dfs(e.next(ve[i]), target);
            if (other.valid()) return other;
        }
    }

    return Edge();
}

int main() {
    string cmd;

    while(cin >> cmd, cmd!=".") {
        string s1, s2, temp;
        if (cmd == "!") {
            int x, y;
            cin >> x >> s1 >> temp >> y >> s2;
            G[s1].push_back(Edge(s2, x, y));
            G[s2].push_back(Edge(s1, y, x));
        } else if (cmd == "?") {
            cin >> s1 >> temp >> s2;
            V.clear();
            Edge e = dfs(Edge(s1, 1, 1), s2);
            if (e.valid())
                cout << e.a << " " << s1 << " = " << e.b << " " << s2 << endl;
            else
                cout << "? " << s1 << " = ? " << s2 << endl;
        }

    }
}
```

uva/10154.cpp

```cpp
//10154
//Weights and Measures
//Dynamic Programming;Longest Increasing Subsequence
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
```

```cpp
#include <climits>
#include <vector>
#include <algorithm>
#define MAX 10005
using namespace std;

struct Turtle {
    int w,c;
    Turtle() {}
    Turtle(int w, int c) : w(w), c(c) {}
};

bool compare(const Turtle& a, const Turtle& b) {
    return a.c > b.c;
}

vector<Turtle> V;
int T[MAX];
int main() {
    int w, c, k=0;
    T[0] = INT_MAX;

    while(cin >> w >> c) {
        V.push_back(Turtle(w, c-w));
    }
    sort(V.begin(), V.end(), compare);

    for(int i=0; i<V.size(); i++) {
        int w = V[i].w, c = V[i].c;

        for(int j=k; j>=0; j--) {
            int next = min(T[j]-w, c);
            if (next >= T[j+1]) {
                T[j+1] = next;
                k=max(k, j+1);
            }
        }
    }
    cout << k << endl;

    return 0;
}
```

uva/10158.cpp

```cpp
//10158
//War
//Misc;Union-Find
#include <iostream>
#include <map>
#include <string>
#include <cstring>
#include <algorithm>
using namespace std;

int P[20000];

inline int enemy(int v) { return v+10000; }

inline int findset(int v) {
    if (P[v] != -1 && P[v] != v)
        return P[v] = findset(P[v]);
    return v;
}

inline int unionset(int x, int y) {
    int a = findset(x), b = findset(y);
    if (a<b) swap(a,b);
    P[b] = a;
}

int main() {
    memset(P, -1, sizeof(P));
    int n, c, x, y;
    cin >> n;
    while(cin >> c >> x >> y, c|x|y) {
        if (c==1) {
            if (findset(x) == findset(enemy(y))) { cout << -1 << endl; continue; }
            unionset(x, y);
            unionset(enemy(x), enemy(y));
        } else if (c==2) {
```

```
            if (findset(x) == findset(y)) { cout << -1 << endl; continue; }
            unionset(x, enemy(y));
            unionset(enemy(x), y);
        } else if (c==3) {
            cout << (findset(x) == findset(y)) << endl;
        } else if (c==4) {
            cout << (findset(x) == findset(enemy(y))) << endl;
        }
    }
}
```

uva/10189.cpp

```
//10189
//Minesweeper
//Misc;Ad hoc
#include <iostream>
#include <cstring>
using namespace std;

char T[200][200];
int N[200][200];
int n, m, t=0;

void add(int i, int j) {
    if (i<0 || i>=n || j<0 || j>=m) return;
    N[i][j]++;
}

int main() {
    while(cin >> n >> m, n|m) {
        memset(N, 0, sizeof(N));
        for(int i=0;i<n;i++) {
            for(int j=0;j<m;j++) {
                cin >> T[i][j];
                if (T[i][j] == '*') {
                    add(i-1, j-1);
                    add(i-1, j);
                    add(i-1, j+1);
                    add(i, j-1);
                    add(i, j+1);
                    add(i+1, j-1);
                    add(i+1, j);
                    add(i+1, j+1);
                }
            }
        }

        if (t++>0) cout << endl;
        cout << "Field #" << t << ":" << endl;
        for(int i=0;i<n;i++) {
            for(int j=0;j<m;j++) {
                if (T[i][j] == '*')
                    cout << T[i][j];
                else
                    cout << N[i][j];
            }
            cout << endl;
        }
    }
}
```

uva/10192.cpp

```
//10192
//Vacation
//Dynamic Programming;Longest Common Subsequence
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
string P, Q;

int main() {
    int p, q, tt=0;
```

```
    while(getline(cin, P), P!="#") {
        tt++;
        getline(cin, Q);
        int p = P.size(), q = Q.size();

        memset(T, 0, sizeof(T));

        for(int i=0; i<=p; i++) T[i][0] = 0;
        for(int i=0; i<=q; i++) T[0][i] = 0;

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                if (P[i-1] == Q[j-1])
                    T[i][j] = T[i-1][j-1] + 1;
                else
                    T[i][j] = max(T[i-1][j], T[i][j-1]);
            }
        }
        cout << "Case #" << tt << ": you can visit at most " << T[p][q] << " cities." << endl;
    }

    return 0;
}
```

uva/10199.cpp

```
//10199
//Tourist Guide
//Graphs;Finding Articulation Points
#include <iostream>
#include <cstring>
#include <map>
#include <vector>
#include <algorithm>
#define MAX 1001
using namespace std;
int G[MAX][MAX], V[MAX], L[MAX], P[MAX], n, m, gpe;
map<string, int> S;
string SR[MAX];
vector<string> F;

void dfs(int u, int v) {
    V[v] = L[v] = ++gpe;
    for(int i = 1; i <= n; i++) {
        if(G[v][i]) {
            if(!V[i]){
                dfs(v, i);
                L[v] = min(L[v], L[i]);
                if(L[i] >= V[v])
                    P[v]++;
            } else if(i != u) {
                L[v] = min(L[v], V[i]);
            }
        }
    }
}

int main() {
    int tt = 0;
    while(cin >> n, n) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        memset(L, 0, sizeof(L));
        memset(P, 0, sizeof(P));
        S.clear();
        F.clear();
        gpe = 0;

        for(int i=1; i<=n; i++) {
            string s; cin >> s;
            S[s] = i;
            SR[i] = s;
        }

        cin >> m;
        for(int i=0; i<m; i++) {
            string s1, s2; cin >> s1 >> s2;
            int a = S[s1], b = S[s2];
            G[a][b] = G[b][a] = 1;
        }

```

```
        for(int i=1; i<=n; i++) {
            if (!V[i]) {
                dfs(i, i);
                P[i]--;
            }
        }

        for(int i=1; i<=n; i++)
            if (P[i]>0)
                F.push_back(SR[i]);

        sort(F.begin(), F.end());

        if (tt) cout << endl;

        cout << "City map #" << ++tt << ": " << F.size() << " camera(s) found" << endl;
        for(int i=0; i<F.size(); i++)
            cout << F[i] << endl;
    }
}
```

uva/10243.cpp

```
//10243
//Fire! Fire! Fire!
//Graphs;DFS
#include <iostream>
#include <iomanip>
#include <cstring>
#include <queue>
#include <cmath>
#define MAX 1005
using namespace std;

int n;
bool G[MAX][MAX];
bool L[MAX];
bool V[MAX];

void dfs(int v, bool start) {
    //cout << "*" << v << endl;
    if (V[v]) return;
    V[v] = true;
    bool all = true;
    int children = 0;
    for(int i=0;i<n;i++) {
        if (G[v][i] && !V[i]) {
            dfs(i, false);
            all &= L[i];
            children++;
        }
    }
    if (!all && children > 0 || start && children==0)
        L[v] = true;
}

int main() {
    int a, b, m;

    while(cin >> n, n) {
        memset(G, 0, sizeof(G));
        memset(L, 0, sizeof(L));
        memset(V, 0, sizeof(V));

        for(int i=0;i<n;i++) {
            cin >> m;
            for(int j=0; j<m; j++) {
                cin >> a;
                a--;
                G[i][a] = G[a][i] = true;
            }
        }

        int count = 0;
        for(int i=0;i<n;i++) {
            dfs(i, true);
            if (L[i]) count++;
        }

        cout << count << endl;
    }
```

```
59
60
61
62  }
```

uva/10259.cpp

```
1   //10259
2   //Hippity Hopscotch
3   //Graphs;DFS
4   #include <iostream>
5   #include <string>
6   #include <cstring>
7   #include <cmath>
8   #include <climits>
9   #define MAX 101
10  using namespace std;
11
12  int T[MAX][MAX], M[MAX][MAX], n, k;
13
14  int walk(int x, int y, int curr) {
15      if (x < 0 || x >= n || y < 0 || y >= n) return 0;
16      if (T[x][y] <= curr) return 0;
17      if (M[x][y] >= 0) return M[x][y];
18
19      int maxx = 0;
20      for(int i=1; i<=k; i++) {
21          maxx = max(maxx, walk(x-i, y, T[x][y])+T[x][y]);
22          maxx = max(maxx, walk(x+i, y, T[x][y])+T[x][y]);
23          maxx = max(maxx, walk(x, y-i, T[x][y])+T[x][y]);
24          maxx = max(maxx, walk(x, y+i, T[x][y])+T[x][y]);
25      }
26      return M[x][y] = maxx;
27  }
28
29  int main() {
30      int t;
31      cin >> t;
32      while(t--) {
33          cin >> n >> k;
34          memset(M, -1, sizeof(M));
35          for(int i=0; i<n; i++)
36              for(int j=0; j<n; j++)
37                  cin >> T[i][j];
38
39          cout << walk(0,0, -1) << endl;
40          if (t) cout << endl;
41      }
42
43      return 0;
44  }
```

uva/10278.cpp

```
1   //10278
2   //Fire Station
3   //Graphs;Shortest Path;Floyd-Warshall
4   #include <iostream>
5   #include <iomanip>
6   #include <cstring>
7   #include <string>
8   #include <sstream>
9   #include <cmath>
10  #include <climits>
11  #define MAX 502
12  using namespace std;
13
14  int G[MAX][MAX], f, n;
15  bool F[MAX];
16
17  int main() {
18      int t; cin >> t;
19      string s;
20      while(t--) {
21          cin >> f >> n;
22          memset(G, 0x3F, sizeof(G));
23          memset(F, 0, sizeof(F));
24
25          for(int i=0;i<f; i++) {
```

```
            int a; cin >> a; F[a] = true;
        }
        getline(cin, s);
        while(getline(cin, s), cin && s!="") {
            int a, b, c;
            stringstream inter(s);
            inter >> a >> b >> c;
            G[a][b] = G[b][a] = c;
        }


        for(int k=1; k<=n; k++) {
            G[k][k] = 0;
            for(int i=1; i<=n; i++)
                for(int j=1; j<=n; j++)
                    G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
        }

        int minn = INT_MAX, minv;
        for(int i=1; i<=n; i++) {
            int maxx = 0;
            for(int j=1; j<=n; j++) {
                int nearest = INT_MAX;
                for(int k=1; k<=n; k++) {
                    if (!F[k] && k!=i) continue;
                    nearest = min(nearest, G[k][j]);
                }
                maxx = max(maxx, nearest);
            }
            if (maxx < minn) {
                minn = maxx;
                minv = i;
            }
        }
        cout << minv << endl;
        if (t) cout << endl;
    }
}
```

uva/10298.cpp

```
//10298
//Power Strings
//Misc;String Matching;KMP
#include <iostream>
#include <string>
#include <cstring>
#define MAX 1000010
using namespace std;

int F[MAX];

void kmp_init(string& P) {
    F[0] = 0;  F[1] = 0;
    int i = 1, j = 0;
    while(i<P.size()) {
        if (P[i] == P[j])
            F[++i] = ++j;
        else if (j == 0)
            F[++i] = 0;
        else
            j = F[j];
    }
}

int kmp(string& P, string& T, int start) {
    kmp_init(P);
    int i = start, j = 0;
    int n = T.size(), m = P.size();

    while(i-j <= n-m) {
        while(j < m) {
            if (P[j] == T[i]) {
                i++; j++;
            } else break;
        }
        if (j == m) return i-m;
        else if (j == 0) i++;
        j = F[j];
    }
}
```

```
41
42
43  int main() {
44      string P, T;
45      while(cin >> P, P!=".") {
46          T = P+P;
47          cout << P.size() / kmp(P, T, 1) << endl;
48      }
49  }
```

uva/10300.cpp

```
1   //10300
2   //Ecological Premium
3   //Misc;Ad hoc
4   #include <iostream>
5   using namespace std;
6
7   int main() {
8       int n, f;
9       cin >> n;
10      while(n--) {
11          cin >> f;
12          double total = 0;
13          for(int i=0; i<f; i++) {
14              double a, b, c;
15              cin >> a >> b >> c;
16              total += a*c;
17          }
18          cout << (int)total << endl;
19      }
20  }
```

uva/10304.cpp

```
1   //10304
2   //Optimal Binary Search Tree
3   //Dynamic Programming;Optimal Search Tree
4   #define MAX 252
5   #include <iostream>
6   #include <cstring>
7   #include <climits>
8   using namespace std;
9
10  int T[MAX][MAX], S[MAX], n;
11  bool V[MAX][MAX];
12
13  int TT(int a, int b) {
14      if (b < a) return 0;
15      if (V[a][b]) return T[a][b];
16
17      int minn = INT_MAX;
18      for(int i=a; i<=b; i++)
19          minn = min(minn, TT(a,i-1) + TT(i+1,b) + (S[b]-S[a-1])-(S[i]-S[i-1]));
20
21      V[a][b] = true;
22      return T[a][b] = minn;
23  }
24
25  int main() {
26      while(cin >> n) {
27          memset(V, 0, sizeof(V));
28          S[0] = 0;
29          for(int i=1; i<=n; i++) {
30              cin >> S[i];
31              S[i] += S[i-1];
32          }
33
34          cout << TT(1, n) << endl;
35
36      }
37  }
```

uva/10316.cpp

```
1   //10316
2   //Airline Hub
```

```cpp
//Math;Geometry;Great-Circle Distance
#define PI 3.14159265
#include <iostream>
#include <cmath>
#include <iomanip>
#include <algorithm>
using namespace std;

struct Point {
    double x, y;
    double dx, dy;

    Point() {}
    Point(double x, double y) : x(x), y(y) {
        dx = x/180.0*PI;
        dy = y/180.0*PI;
    }

    double distance(Point& p) {
        return acos(cos(p.dx) * cos(this->dx) * cos(this->dy - p.dy) + sin(p.dx) * sin(this->dx));
    }
};

Point P[1050];

int main() {
    int n;
    while(cin >> n) {
        for(int i=0; i<n; i++) {
            double x,y; cin >> x >> y;
            P[i] = Point(x,y);
        }

        double minn = 1000000000;
        int mini = 0;
        for(int i=0; i<n; i++) {
            double maxx = 0;
            for(int j=0;j<n;j++)
                maxx = max(maxx, P[i].distance(P[j]));

            if (maxx < minn || abs(maxx - minn) < 1e-6) {
                mini = i;
                minn = maxx;
            }
        }

        cout << fixed << setprecision(2) << P[mini].x << " " << P[mini].y << endl;
    }


}
```

uva/10319.cpp

```cpp
//10319
//Manhattan
//Graphs;2-SAT
#include <iostream>
#include <string>
#include <cstring>
#include <vector>
#define MAX 1000
using namespace std;

vector<int> G[MAX*2], T[MAX*2];
int O[MAX*2], V[MAX*2], npv, n, s, a;

int neg(int x) {
    if (x>=n) return x-n;
    return x+n;
}

int av(int x) {
    return s+x;
}

int st(int x) {
    return x;
}

void DFS(int v){
```

```
    V[v] = 1;
    for(int i = 0; i < G[v].size(); i++)
        if (!V[G[v][i]])
            DFS(G[v][i]);
    O[npv++] = v;
}

void DFSt(int v, int comp){
    V[v] = comp;
    for(int i = 0; i < T[v].size(); i++)
       if (!V[T[v][i]])
            DFSt(T[v][i], comp);
}


int main() {
    int m;
    int tt; cin >> tt;
    while(tt--) {
        cin >> s >> a >> m;
        n = (s+a);

        for(int i=0;i<2*n; i++) {
            G[i].clear();
            T[i].clear();
        }

        for(int i=0; i<m; i++) {
            int s1, a1, s2, a2;
            cin >> s1 >> a1 >> s2 >> a2;
            s1--; a1--; s2--; a2--;

            s1 = st(s1); s2=st(s2);
            a1 = av(a1); a2=av(a2);

            if (a1 == a2 && s1 == s2)
                continue;

            if (a2<a1) {
                s1 = neg(s1);
                s2 = neg(s2);
            }

            if (s2<s1) {
                a1 = neg(a1);
                a2 = neg(a2);
            }

            if (a1 == a2) {
                G[neg(a1)].push_back(a1);
                continue;
            }

            if (s1 == s2) {
                G[neg(s1)].push_back(s1);
                continue;
            }

            G[neg(s1)].push_back(a1);
            G[neg(a1)].push_back(s1);

            G[neg(s1)].push_back(s2);
            G[neg(s2)].push_back(s1);

            G[neg(a2)].push_back(a1);
            G[neg(a1)].push_back(a2);

            G[neg(a2)].push_back(s2);
            G[neg(s2)].push_back(a2);
        }

        for(int i=0; i<2*n; i++)
            for(int j=0; j<G[i].size(); j++)
                T[G[i][j]].push_back(i);


        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i = 0; i < 2*n; i++)
            if(!V[i]) DFS(i);
```

```
110
111         memset(V, 0, sizeof(V));
112
113         int comp = 0;
114         for(int i = 2*n-1; i >= 0; i--)
115             if(!V[O[i]])
116                 DFSt(O[i], ++comp);
117
118         bool result = true;
119         for(int i=0; i<n; i++) {
120             result &= V[i] != V[neg(i)];
121         }
122
123         cout << (result ? "Yes" : "No") << endl;
124     }
125 }
```

uva/10389.cpp

```
1  //10389
2  //Subway
3  //Graphs;Shortest Path;Dijkstra
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <vector>
8  #include <algorithm>
9  #include <queue>
10 #include <cmath>
11 #include <sstream>
12 #include <string>
13 #include <iomanip>
14 #include <cassert>
15 #define MAX 205
16 #define WALK 1
17 #define METRO 4
18
19 using namespace std;
20
21 struct Edge {
22     int v; double c;
23     Edge(int v, double c) : v(v), c(c) {}
24     inline bool operator < (const Edge& that) const { return c > that.c; }
25 };
26
27 vector<Edge> G[MAX];
28 double V[MAX];
29 double X[MAX], Y[MAX];
30 int n;
31
32
33 double dist(double ax, double ay, double bx, double by) {
34     return sqrt(pow(ax-bx, 2.0) + pow(ay-by, 2.0))*60/10000;
35 }
36
37 int main() {
38     int t; cin >> t; t=0;
39     while(cin >> X[0] >> Y[0] >> X[1] >> Y[1]) {
40         memset(G, 0, sizeof(G));
41
42         G[0].push_back(Edge(1, dist(X[0], Y[0], X[1], Y[1])));
43         G[1].push_back(Edge(0, dist(X[0], Y[0], X[1], Y[1])));
44
45         n = 2;
46         string s;
47         getline(cin, s);
48         while(getline(cin, s) && s!="" && s[0]!=' ') {
49             stringstream sin(s);
50             int mn=0;
51             while(sin >> X[n] >> Y[n]) {
52                 if (X[n] == -1 && Y[n] == -1) {
53                     assert(mn >= 2);
54                     mn = 0;
55                     break;
56                 }
57                 if (mn > 0) {
58                     double mDist = dist(X[n-1], Y[n-1], X[n], Y[n])/METRO;
59                     G[n-1].push_back(Edge(n, mDist));
60                     G[n].push_back(Edge(n-1, mDist));
61                 }
62                 for(int i=0;i<n;i++) {
```

```
                    double aDist = dist(X[n], Y[n], X[i], Y[i])/WALK;
                    G[i].push_back(Edge(n, aDist));
                    G[n].push_back(Edge(i, aDist));
                }

                n++; mn++;
            }

        }

        int totalc=0;

        for(int i=0; i<n; i++) V[i] = -1;

        priority_queue<Edge> Q;
        Q.push(Edge(0, 0));

        while(totalc < n && !Q.empty()) {
            Edge item = Q.top(); Q.pop();
            if (item.c >= V[item.v] && V[item.v] >= 0) continue;

            V[item.v] = item.c;
            totalc++;

            for(int j=0; j<G[item.v].size(); j++) {
                Edge e = G[item.v][j];
                if (item.c + e.c < V[e.v] || V[e.v] == -1)
                    Q.push(Edge(e.v, item.c + e.c));
            }
        }

        if (t++) cout << endl;
        cout << (int)round(V[1]) << endl;
    }
    return 0;
}
```

uva/10397.cpp

```
//10397
//Connect the Campus
//Graphs;Minimum Spanning Tree;Prim;Priority Queue
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#include <cmath>
#include <iomanip>
#define MAX 200010

using namespace std;

struct Road {
    int v; double c;
    Road(int v, double c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};


vector<Road> G[MAX];
int X[MAX], Y[MAX];
priority_queue<Road> Q;
int n, m;
bool V[MAX];


int main() {
    while(cin >> n) {
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        Q = priority_queue<Road>();

        for(int i=1; i<=n; i++) {
            int x, y;
            cin >> x >> y;
            X[i] = x; Y[i] = y;
            for(int j=1; j<=i; j++) {
                double d = sqrt(pow(X[i]-X[j], 2.0)+pow(Y[i]-Y[j], 2.0));
                G[i].push_back(Road(j, d));
```

```
43                  G[j].push_back(Road(i, d));
44              }
45          }
46
47          cin >> m;
48          for(int i=0; i<m; i++) {
49              int a, b;
50              cin >> a >> b;
51              G[a].push_back(Road(b, 0));
52              G[b].push_back(Road(a, 0));
53          }
54
55          double total = 0; int totalc=0;
56          Q.push(Road(1,0));
57
58          while(totalc < n && !Q.empty()) {
59              Road item = Q.top(); Q.pop();
60              if (V[item.v]) continue;
61
62              V[item.v] = true;
63              total += item.c;
64              totalc++;
65
66              for(int j=0; j<G[item.v].size(); j++)
67                  if (!V[G[item.v][j].v])
68                      Q.push(G[item.v][j]);
69          }
70
71          cout << setprecision(2);
72          cout << fixed << total << endl;
73      }
74      return 0;
75  }
```

uva/10420.cpp

```
1   //10420
2   //List of Conquests
3   //Misc;STL map
4   #include <iostream>
5   #include <string>
6   #include <cstring>
7   #include <cmath>
8   #include <map>
9   #define MAX 105
10  using namespace std;
11
12  map<string, int> women;
13  int main() {
14      int n;
15      string s;
16      cin >> n;
17      while(n--) {
18          cin >> s;
19          women[s]++;
20          getline(cin, s);
21      }
22
23      for(map<string, int>::const_iterator it = women.begin(); it != women.end(); it++) {
24          cout << it->first << " " << it->second << endl;
25      }
26
27      return 0;
28  }
```

uva/10444.cpp

```
1   //10444
2   //Multi-peg Towers of Hanoi
3   //Dynamic Programming;Ad hoc
4   #include <iostream>
5   #include <string>
6   #include <cstring>
7   #include <cmath>
8   #include <climits>
9   #define MAX 205
10  using namespace std;
11
12  int T[MAX][MAX];
```

```
int main() {
    int n=201, p=21, t=0;

    for(int i=0;i<=n;i++) {
        if (i<31)
            T[i][3] = (1<<i)-1;
        else
            T[i][3] = INT_MAX; //avoid overflow
    }

    for(int i=1; i<=n; i++) {
        for(int j=4; j<=p; j++) {
            if (i<j) {
                T[i][j] = 2*i-1;
            } else {
                int minn = INT_MAX;
                for(int k=1;k<i;k++) {
                    int value = 2*T[k][j]+T[i-k][j-1];
                    if (value >= 0) //avoid overflow
                        minn = min(minn, value);
                }
                T[i][j] = minn;
            }

        }
    }

    while(cin >> n >> p, n | p) {
        cout << "Case " << ++t << ": " << T[n][p] << endl;
    }

    return 0;
}
```

uva/10462.cpp

```
//10462
//Is There A Second Way Left?
//Graphs;Minimum Spanning Tree;Kruskal
#include <iostream>
#include <cstring>
#include <vector>
#include <algorithm>
#include <cassert>
using namespace std;

struct Edge {
    int x, y, v;
    inline bool operator <(const Edge& that) const {
        return this->v < that.v;
    }
};

Edge E[205];
int P[105], S[105];

inline int findset(int v) {
    if (P[v] != v)
        return P[v] = findset(P[v]);
    return v;
}

inline int unionset(int x, int y) {
    int a = findset(x), b = findset(y);
    if (a==b) return -1;
    if (a>b) swap(a,b);
    P[b] = a;
    return a;
}

int best(int n, int m, int skip) {
    for(int i=0; i<=n; i++)
        P[i] = i;

    int total=0, count=0;
    for(int i=0; i<m && count < n-1; i++) {
        if(i!=skip && unionset(E[i].x, E[i].y) != -1) {
            total += E[i].v;
            if (skip == -1)
                S[count] = i;
            count++;
```

```
        }
    }
    if (count == n-1)
        return total;
    else
        return -1;
}

int main() {
    int t; cin >> t;
    for(int tt=1; tt<=t; tt++) {
        int n, m; cin >> n >> m;

        for(int i=0; i<m; i++)
            cin >> E[i].x >> E[i].y >> E[i].v;

        sort(E, E+m);

        cout << "Case #" << tt << " : ";

        if (best(n, m, -1) == -1) {
            cout << "No way" << endl;
            continue;
        }

        int minn = 1<<30;
        for(int i=0;i<n-1; i++) {
            int value = best(n, m, S[i]);
            if (value != -1)
                minn = min(minn, value);
        }


        if (minn < 1<<30)
            cout << minn << endl;
        else
            cout << "No second way" << endl;
    }
}
```

uva/10480.cpp

```
//10480
//Sabotage
//Graphs;Maximum Flow;Ford-Fulkerson
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#include <climits>
#define MAX 1006
using namespace std;

int G[MAX][MAX], O[MAX][MAX], n, m;
bool V[MAX];

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=1; i<=n; i++) {
        if (!V[i] && G[s][i] > 0) {
            if (int sent = send(i, t, min(minn, G[s][i]))) {
                G[s][i] -= sent;
                G[i][s] += sent;
                return sent;
            }
        }
    }
    return 0;
}

int main() {
    int tt=0;
    while(cin >> n >> m, n|m) {
        if (tt++) cout << endl;

        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));
```

```
40
41          for(int i=0;i<m;i++) {
42              int a, b, f;
43              cin >> a >> b >> f;
44              G[a][b] = G[b][a] += f;
45              O[a][b] += f;
46          }
47
48          int total = 0;
49          while(int sent = send(1, 2, INT_MAX)) {
50              total += sent;
51              memset(V, 0, sizeof(V));
52          }
53          for(int i=1;i<=n;i++) {
54              for(int j=1;j<=n;j++) {
55                  if (O[i][j] > 0 && V[i] != V[j])
56                      cout << i << " " << j << endl;
57              }
58          }
59      }
60  }
```

uva/10511.cpp

```
1   //10511
2   //Councilling
3   //Graphs;Maximum Flow;Ford-Fulkerson
4   #include <iostream>
5   #include <iomanip>
6   #include <cstring>
7   #include <sstream>
8   #include <string>
9   #include <cmath>
10  #include <map>
11  #include <climits>
12  #define MAX 1300
13  using namespace std;
14
15  int G[MAX][MAX], n;
16  bool V[MAX];
17  map<string, int> EC, EP, EM;
18
19  int SOURCE() { return 1; }
20  int P(string& s) { if (EP.find(s)!=EP.end()) return EP[s]; else { return EP[s]=++n;} }
21  int M(string& s) { if (EM.find(s)!=EM.end()) return EM[s]; else { return EM[s]=++n;} }
22  int C(string& s) { if (EC.find(s)!=EC.end()) return EC[s]; else { return EC[s]=++n;} }
23  int TARGET() { return 2; }
24
25  int send(int s, int t, int minn) {
26      V[s] = true;
27
28      if (s==t) return minn;
29      for(int i=1; i<=n; i++) {
30          if (!V[i] && G[s][i] > 0) {
31              if (int sent = send(i, t, min(minn, G[s][i]))) {
32                  G[s][i] -= sent;
33                  G[i][s] += sent;
34                  return sent;
35              }
36          }
37      }
38      return 0;
39  }
40
41  int main() {
42      int t; cin >> t;
43      string s, sm, sp, sc;
44      getline(cin, s); getline(cin, s);
45      while(t--) {
46          EC.clear(); EP.clear(); EM.clear();
47          memset(G, 0, sizeof(G));
48          memset(V, 0, sizeof(V));
49          n=2;
50
51          while(getline(cin, s) && s!="" && s!=" ") {
52              stringstream sin(s);
53              sin >> sm >> sp;
54              G[P(sp)][M(sm)] = 1;
55              while(sin >> sc) {
56                  G[M(sm)][C(sc)] = 1;
57                  G[C(sc)][TARGET()] = 1;
```

```
            }
        }

        int maxParty = (EC.size()-1)/2;
        for(map<string, int>::iterator it=EP.begin(); it!=EP.end(); it++) {
            G[SOURCE()][it->second] = maxParty;
        }


        int total = 0;
        while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }

        if (total == EC.size()) {
            for(map<string, int>::iterator i=EM.begin(); i!=EM.end(); i++) {
                for(map<string, int>::iterator j=EC.begin(); j!=EC.end(); j++) {
                    if (G[j->second][i->second]) {
                        cout << i->first << " " << j->first << endl;
                    }
                }
            }
        } else {
            cout << "Impossible." << endl;
        }

        if (t) cout << endl;
    }
}
```

uva/10557.cpp

```
//10557
//XYZZY
//Graphs;Shortest Path;Bellman Ford
#include <iostream>
#include <cstring>
#include <vector>
#include <algorithm>
#define MAX 1001
using namespace std;

struct Edge {
    int a, b;
    Edge(int a, int b) : a(a), b(b) {}
};

int X[MAX], M[MAX], V[MAX], n;
vector<Edge> E;

bool reach(int v, int t) {
    if (v==t) return true;
    V[v] = true;
    for(int i=0; i<E.size(); i++) {
        Edge e = E[i];
        if (e.a == v && !V[e.b] && reach(e.b, t))
            return true;
    }
    return false;
}

int main() {
    while(cin >> n, n!=-1) {
        memset(V, 0, sizeof(V));
        E.clear();
        for(int a=1; a<=n; a++) {
            int k; cin >> X[a] >> k;

            for(int j=0; j<k; j++) {
                int b; cin >> b;
                E.push_back(Edge(a, b));
            }
        }

        M[1] = 100;
        for(int a=2; a<=n; a++)
            M[a] = -1<<29;

        for(int k=0; k<n-1; k++) {
            for(int i=0; i<E.size(); i++) {
```

```
                Edge e = E[i];
                if (M[e.a]<=0) continue;
                M[e.b] = max(M[e.b], M[e.a] + X[e.b]);
            }
        }

        bool cycle = false;
        for(int i=0; i<E.size(); i++) {
            Edge e = E[i];
            if (M[e.a]<=0) continue;
            cycle |= M[e.a]+X[e.b] > M[e.b] && reach(e.a, n);
        }


        if (M[n] > 0 || cycle) {
            cout << "winnable" << endl;
        } else {
            cout << "hopeless" << endl;
        }

    }
}
```

uva/10594.cpp

```
//10594
//Data Flow
//Graphs;Maximum Flow;Min Cost;Cycle Canceling
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#define MAX 110
using namespace std;

struct Item {
    long long v, p, c;
    Item() {}
    Item(long long v, long long p, long long c) : v(v), p(p), c(c) {}
};

long long C[MAX][MAX], M[MAX];
long long F[MAX][MAX], G[MAX][MAX], P[MAX], n, a, b, qn;
bool V[MAX];
Item Q[MAX];


long long send(long long s, long long t) {
    memset(V, 0, sizeof(V));
    qn = 0;

    Q[qn++] = Item(s, -1, 1L<<60);
    V[s] = true;

    for(long long i=0; i<qn; i++) {
        Item item = Q[i];

        if (item.v == t) {
            long long sent = item.c;
            while(item.p != -1) {
                Item parent = Q[item.p];
                F[parent.v][item.v] += sent;
                F[item.v][parent.v] -= sent;
                item = parent;
            }

            return sent;
        }

        for(long long j=0; j<n; j++) {
            long long residual = G[item.v][j]-F[item.v][j];
            if (!V[j] && residual) {
                V[j] = true;
                Q[qn++] = Item(j, i, min(item.c, residual));
            }
        }
    }

    return 0;
}
```

```
57
58  bool cancel_cycle(long long source) {
59      memset(M, 0x1f, sizeof M);
60      M[source] = 0;
61
62      bool cycle = false;
63      long long v;
64
65      for(long long k=0; k<n; k++)
66      for(long long i=0; i<n; i++)
67      for(long long j=0; j<n; j++) {
68          if (G[i][j] - F[i][j] && M[i]+C[i][j]<M[j]) {
69              cycle = k+1==n;
70              v = i;
71
72              M[j] = M[i] + C[i][j];
73              P[j] = i;
74          }
75
76          if (F[i][j]>0 && M[j]-C[i][j]<M[i]) {
77              cycle = k+1==n;
78              v = j;
79
80              M[i] = M[j] - C[i][j];
81              P[i] = j;
82          }
83      }
84
85      if (not cycle) return false;
86
87      for(long long k=0; k<n; k++)
88          v = P[v];
89
90      long long i=v, minn=1L<<60;
91
92      do {
93          if (F[P[i]][i] < 0) {
94              minn = min(minn, -F[P[i]][i]);
95          } else {
96              minn = min(minn, G[P[i]][i] - F[P[i]][i]);
97          }
98      } while (i = P[i], i!=v);
99
100     do {
101         F[P[i]][i] += minn;
102         F[i][P[i]] -= minn;
103     } while (i = P[i], i!=v);
104
105     return true;
106 }
107
108 int main() {
109     long long m;
110     while(cin >> n >> m) {
111         n++;
112         memset(G, 0, sizeof(G));
113         memset(F, 0, sizeof(F));
114         memset(C, 0, sizeof(C));
115
116         for(long long i=0; i<m; i++) {
117             long long x, y, c;
118             cin >> x >> y >> c;
119
120             C[x][y] = C[y][x] = c;
121             G[x][y] = G[y][x] = 1;
122         }
123         long long d, k; cin >> d >> k;
124         G[0][1] = d;
125         G[1][0] = d;
126
127         for(long long i=1; i<n; i++) {
128             for(long long j=1; j<n; j++) {
129                 G[i][j] *= k;
130             }
131         }
132
133         long long total = 0;
134         while(long long sent = send(0, n-1))
135             total += sent;
136
137         while(cancel_cycle(0));
138
```

```
        long long cost = 0;
        for(long long i=0; i<n; i++)
            for(long long j=0; j<n; j++)
                if (F[i][j] > 0) {
  //                    cout << " " << i << " " << j << " " << F[i][j] << " " << C[i][j] << endl;
                    cost += F[i][j] * C[i][j];
                } else if (F[i][j] < 0) {
//                    cout << " " << i << " " << j << " " << F[i][j] << " " << C[i][j] << endl;

                }

        if (total != d)
            cout << "Impossible." << endl;
        else
            cout << cost << endl;
    }
}
```

uva/10635.cpp

```
//10635
//Prince and Princess
//Dynamic Programming;Longest Increasing Subsequence
#include <iostream>
#include <cstring>
#include <climits>
#include <cmath>
#include <algorithm>
#define MAX 255*255
using namespace std;

int P[MAX], Q[MAX], M[MAX];

int main() {
    int n, p, q, tt=0, temp;
    cin >> n;
    while(cin >> n >> p >> q) {
        memset(P, 0, sizeof(P));
        q++; p++;
        for(int i=1;i<=p;i++) {
            cin >> temp;
            P[temp] = i;
        }

        for (int i=1;i<=q;i++) {
            cin >> temp;
            Q[i] = P[temp];
        }

        int k=0; M[0]=0;
        for(int i=1;i<=q;i++) {
            if (Q[i] > M[k]) {
                k++; M[k] = Q[i];
            } else {
                int j = (int)(lower_bound(M, M+k+1, Q[i])-M);
                if (Q[i] > M[j]) j++;
                M[j] = Q[i];
            }
        }

        cout << "Case " << ++tt << ": " << k << endl;
    }

    return 0;
}
```

uva/10652.cpp

```
//10652
//Board Wrapping
//Math;Geometry;Convex Hull;Monotone Chain
#include <iostream>
#include <cmath>
#include <iomanip>
#include <algorithm>
using namespace std;

double PI = 2*acos(0.0);

```

```
12  struct Point {
13      double x, y;
14
15      Point() {}
16      Point(double x, double y) : x(x), y(y) {}
17
18      double product(Point a, Point b) {
19          return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
20      }
21
22      bool right(Point a, Point b) {
23          return product(a, b) > 0;
24      }
25
26      double dist(Point b) {
27          return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y));
28      }
29
30      bool operator <(const Point& p) const {
31          if (this->x != p.x) return this->x < p.x;
32          return this->y < p.y;
33      }
34
35      bool operator ==(const Point& p) const {
36          return this->x == p.x and this->y == p.y;
37      }
38
39      Point rotateWith(const Point origin, double si, double co, double scale) const {
40          double tx = this->x - origin.x;
41          double ty = this->y - origin.y;
42          double x = (tx * co + ty * si)/scale;
43          double y = (tx * -si + ty * co)/scale;
44          return Point(origin.x + x, origin.y + y);
45
46      }
47  };
48
49  int convexHull(Point* P, int n, Point* S) {
50      sort(P, P+n);
51
52      int m=0;
53      for(int i=0; i<n; i++) {
54          while(m >= 2 && !S[m-1].right(S[m-2], P[i])) m--;
55          S[m++] = P[i];
56      }
57      m--;
58
59      for(int i=n-1, k=m; i >= 0; i--) {
60          while(m >= k+2 && !S[m-1].right(S[m-2], P[i])) m--;
61          S[m++] = P[i];
62      }
63      m--;
64
65      return m;
66  }
67
68  double area(Point* A, int a) {
69      double area = 0;
70      for(int i=0; i<a; i++) {
71          int j = (i+1)%a;
72          area += (A[i].x + A[j].x) * (A[i].y - A[j].y);
73      }
74      return area / 2;
75  }
76
77  Point P[3000], S[3000];
78
79  int main() {
80      int tt; cin >> tt;
81      while(tt--) {
82          int n; cin >> n;
83
84          double initialArea = 0;
85          for(int i=0; i<n; i++) {
86              Point p;
87              cin >> p.x >> p.y;
88              double w, h, angle;
89              cin >> w >> h >> angle;
90
91              angle *= PI/180;
92
93              Point a = Point(p.x-w/2, p.y-h/2).rotateWith(p, sin(angle), cos(angle), 1);
```

```
            Point b = Point(p.x-w/2, p.y+h/2).rotateWith(p, sin(angle), cos(angle), 1);
            Point c = Point(p.x+w/2, p.y+h/2).rotateWith(p, sin(angle), cos(angle), 1);
            Point d = Point(p.x+w/2, p.y-h/2).rotateWith(p, sin(angle), cos(angle), 1);

            P[i*4+0] = a;
            P[i*4+1] = b;
            P[i*4+2] = c;
            P[i*4+3] = d;

            initialArea += area(P+i*4, 4);
        }

        int s = convexHull(P, n*4, S);
        double finalArea = area(S, s);

        cout << fixed << setprecision(1) << abs(100*(initialArea/finalArea)) << " %" << endl;
    }
}
```

uva/10684.cpp

```
//10684
//The Jackpot
//Dynamic Programming;Maximum Sum Contiguous Subsequence
#include <iostream>
#include <cmath>
#define MAX 1005
using namespace std;

int main() {
    int n, a;
    while(cin >> n, n) {
        int t=0, s=0;
        for(int i=0;i<n;i++) {
            cin >> a;
            if (s+a>=0)
                t = max(t, s+=a);
            else
                s = 0;
        }
        if (s>0) {
            cout << "The maximum winning streak is " << t << "." << endl;
        } else {
            cout << "Losing streak." << endl;
        }
    }

    return 0;
}
```

uva/10696.cpp

```
//10694
//f91
//Misc;Ad hoc
#include <iostream>
using namespace std;

int f91(int x) {
    if (x<=100)
        return f91(f91(x+11));
    else
        return x-10;
}

int main() {
    int n;
    while(cin >> n, n)
        cout << "f91(" << n << ") = " << f91(n) << endl;
}
```

uva/10723.cpp

```
//10723
//Cyborg Genes
//Dynamic Programming;Longest Common Subsequence
#include <iostream>
```

```cpp
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX], D[MAX][MAX];
string P, Q;

int main() {
    int p, q, t, tt=0;
    cin >> t;
    getline(cin, P);
    while(tt++ < t) {
        getline(cin, P);
        getline(cin, Q);
        int p = P.size(), q = Q.size();

        for(int i=0; i<=p; i++) { T[i][0] = 0; D[i][0] = 1; }
        for(int i=0; i<=q; i++) { T[0][i] = 0; D[0][i] = 1; }

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                D[i][j] = 0;
                if (P[i-1] == Q[j-1]) {
                    T[i][j] = T[i-1][j-1] + 1;
                    D[i][j] = D[i-1][j-1];
                }
                else {
                    T[i][j] = max(T[i-1][j], T[i][j-1]);
                    if (T[i-1][j] == T[i][j]) D[i][j] += D[i-1][j];
                    if (T[i][j-1] == T[i][j]) D[i][j] += D[i][j-1];
                }
            }
        }

        cout << "Case #" << tt << ": " << p+q-T[p][q] << " " << D[p][q] << endl;
    }

    return 0;
}
```

uva/10724.cpp

```cpp
//10724
//Road Construction
//Graphs;Shortest Path;Floyd-Warshall
#include <iostream>
#include <cmath>
#define MAX 58
#define EP 1e-6
#define eq(x, y) abs(x-y) < EP
using namespace std;

double G[MAX][MAX], X[MAX], Y[MAX];

double dist(int i, int j) {
    return sqrt(pow(X[i]-X[j], 2.0) + pow(Y[i]-Y[j], 2.0));
}

struct Answer {
    double x, d;
    int i, j;
    Answer(double x, double d, int i, int j) : x(x), d(d), i(i), j(j) {}

    bool operator <(const Answer& o) const{
        if (!eq(x, o.x)) return x < o.x;
        if (!eq(d, o.d)) return d > o.d;
        if (i!=o.i) return i>o.i;
        if (j!=o.j) return j>o.j;
        return false;
    }

    bool valid() {
        return x-1 > EP;
    }
};

int main() {
    int n, m;
    while(cin >> n >> m, n|m) {
```

```
        for(int i=1; i<=n; i++) {
            for(int j=1; j<=n; j++)
                G[i][j] = 1e8;

            G[i][i] = 0;
            cin >> X[i] >> Y[i];
        }

        for(int i=0; i<m; i++) {
            int x, y;
            cin >> x >> y;
            G[x][y] = G[y][x] = dist(x, y);
        }

        for(int k=1; k<=n; k++)
            for(int i=1; i<=n; i++)
                for(int j=1; j<=n; j++)
                    G[i][j] = min(G[i][j], G[i][k] + G[k][j]);

        Answer maxx(0, 0, 0, 0);

        for(int u=1; u<=n; u++) {
            for(int v=1; v<=n; v++) {
                double improve = 0, uv = dist(u, v);

                for(int i=1; i<=n; i++)
                    for(int j=1; j<=n; j++)
                        improve += G[i][j] - min(G[i][j],
                            min(G[i][u]+uv+G[v][j], G[i][v]+uv+G[u][j]));

                maxx = max(maxx, Answer(improve, uv, u, v));
            }
        }

        if (maxx.valid()) {
            cout << maxx.i << " " << maxx.j << endl;
        } else {
            cout << "No road required" << endl;
        }
    }
}
```

uva/10739.cpp

```
//10739
//String to Palindrome
//Dynamic Programming;Edit Distance
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
string P, Q;

int main() {
    int p, q, t, tt=0;
    cin >> t;
    getline(cin, P);
    while(tt++ < t) {
        getline(cin, P);
        Q = string(P.rbegin(), P.rend());
        int p = P.size(), q = Q.size();

        for(int i=0; i<=p; i++) { T[i][0] = i; }
        for(int i=0; i<=q; i++) { T[0][i] = i; }

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                if (P[i-1] == Q[j-1])
                    T[i][j] = T[i-1][j-1];
                else
                    T[i][j] = min(min(T[i-1][j], T[i][j-1]), T[i-1][j-1])+1;
            }
        }

        cout << "Case " << tt << ": " << T[p][q]/2 << endl;
    }

```

```
    return 0;
}
```

uva/10746.cpp

```
//10746
//Crime Wave - The Sequel
//Graphs;Maximum Flow;Min Cost;Cycle Canceling
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#define MAX 100
using namespace std;

double C[MAX][MAX], M[MAX];
int F[MAX][MAX], G[MAX][MAX], P[MAX], n, a, b;
bool V[MAX];

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;

    for(int i=0; i<n; i++) {
        if (!V[i] && G[s][i]-F[s][i]) {
            if (int sent = send(i, t, min(minn, G[s][i]-F[s][i]))) {
                F[s][i] += sent;
                F[i][s] -= sent;
                return sent;
            }
        }
    }

    return 0;
}

bool cancel_cycle(int source) {
    memset(M, 0x1f, sizeof M);
    M[source] = 0;

    bool cycle = false;
    int v;

    for(int k=0; k<n; k++)
    for(int i=0; i<n; i++)
    for(int j=0; j<n; j++)
        if (G[i][j]-F[i][j] && M[i]+C[i][j]<M[j]) {
            cycle = k+1==n;
            v = i;

            M[j] = M[i] + C[i][j];
            P[j] = i;
        }

    if (not cycle) return false;

    for(int k=0; k<n; k++)
        v = P[v];

    int i=v, minn=1<<29;

    do {
        minn = min(minn, G[P[i]][i] - F[P[i]][i]);
    } while (i = P[i], i!=v);

    do {
        F[P[i]][i] += minn;
        F[i][P[i]] -= minn;
    } while (i = P[i], i!=v);

    return true;
}

int cruiser(int x) { return x; }
int bank(int x) { return b+x; }
int source() { return a+b; }
int target() { return a+b+1; }

int main() {
```

```
    while(cin >> a >> b, a|b) {
        memset(G, 0, sizeof(G));
        memset(F, 0, sizeof(F));
        memset(C, 0, sizeof(C));

        for(int i=0; i<a; i++)
            G[bank(i)][target()] = 1;

        for(int i=0; i<b; i++)
            G[source()][cruiser(i)] = 1;

        for(int i=0; i<a; i++) {
            for(int j=0; j<b; j++) {
                int cr = cruiser(j), bk = bank(i);
                cin >> C[cr][bk];
                G[cr][bk] = 1;
                C[bk][cr] = -C[cr][bk];
            }
        }

        n = target()+1;

        int total = 0, sent;
        while(memset(V, 0, sizeof V), sent = send(source(), target(), 1<<29))
            total += sent;

        while(cancel_cycle(source()));

        double cost = 0;
        for(int i=0; i<n; i++)
            for(int j=0; j<n; j++)
                if (F[i][j] > 0)
                    cost += F[i][j] * C[i][j];

        cout << fixed << setprecision(2) << cost/a+1e-6 << endl;
    }
}
```

uva/10783.cpp

```
//10783
//Odd Sum
//Misc;Ad hoc
#include <iostream>
#include <cstring>
#include <cmath>
using namespace std;

int main() {
    int t;
    cin >> t;
    for(int tt=1;tt<=t;tt++) {
        int a, b;
        cin >> a >> b;
        int s = 0;
        for(int i=a;i<=b;i++) {
            if (i&1)
                s+=i;
        }
        cout << "Case " << tt << ": " << s << endl;

    }
}
```

uva/10793.cpp

```
//10793
//The Orc Attack
//Graphs;Shortest Path;Floyd-Warshall
#include <iostream>
#include <algorithm>
#define MAX 105
using namespace std;

int G[MAX][MAX];

int main() {
    int t; cin >> t;
    for(int tt=1; tt<=t; tt++) {
```

```
        int n, m; cin >> n >> m;
        for(int i=1; i<=n; i++) {
            for(int j=1; j<=n; j++)
                G[i][j] = 1<<29;
            G[i][i] = 0;
        }

        for(int i=0; i<m; i++) {
            int x, y, c;
            cin >> x >> y >> c;
            G[x][y] = G[y][x] = min(G[x][y], c);
        }

        for(int k=1; k<=n; k++)
            for(int i=1; i<=n; i++)
                for(int j=1; j<=n; j++)
                    G[i][j] = min(G[i][j], G[i][k] + G[k][j]);

        int minn = 1<<29;
        for(int i=1; i<=n; i++)
            if (*min_element(G[i]+1, G[i]+6) == *max_element(G[i]+1, G[i]+6))
                minn = min(minn, *max_element(G[i]+1, G[i]+n+1));

        cout << "Map " << tt << ": ";
        if (minn < 1<<29)
            cout << minn << endl;
        else
            cout << -1 << endl;
    }
}
```

uva/10827.cpp

```
//10827
//Maximum sum on a torus
//Dynamic Programming;Maximum Sum Sub-rectangle
#include <iostream>
#include <climits>
#define MAX 160
using namespace std;

int T[MAX][MAX];

int main() {
    int n, a, cases;
    cin >> cases;
    while(cin >> n) {
        for(int i=1; i<=n; i++) {
            for(int j=1;j<=n;j++) {
                cin >> T[i][j];
                T[i+n][j] = T[i][j];
            }
        }

        for(int i=1; i<=2*n; i++)
            for(int j=1;j<=n; j++)
                T[i][j]+=T[i-1][j];

        int t = 0;
        for(int i=1;i<=2*n; i++) {
            for(int j=i;j<=min(i+n-1, 2*n);j++) {
                int smax=0, smin=0, ssum=0, tmax=0, tmin=0;
                for(int k=1;k<=n; k++)
                    ssum += T[j][k] - T[i-1][k];

                for(int k=1;k<=n; k++) {
                    int a = T[j][k] - T[i-1][k];
                    smax += a;
                    smin += a;

                    tmax = max(tmax, smax);
                    tmin = min(tmin, smin);

                    if (smax < 0) smax = 0;
                    if (smin > 0) smin = 0;
                }
                t = max(t, max(tmax, ssum-tmin));
            }
        }

        cout << t << endl;
```

```
    }

    return 0;
}
```

uva/10891.cpp

```
//10891
//Game of Sum
//Dynamic Programming;Matrix Multiplication
#define MAX 101
#include <iostream>
#include <cstring>
#include <climits>
using namespace std;

int T[MAX][MAX], S[MAX], n;
bool V[MAX][MAX];

int TT(int a, int b) {
    if (b<a) return 0;
    if (V[a][b]) return T[a][b];

    int maxx = INT_MIN;
    for(int i=a; i<=b; i++)
        maxx = max(maxx, S[b]-S[a-1] - TT(i+1,b));

    for(int i=b; i>=a; i--)
        maxx = max(maxx, S[b]-S[a-1] - TT(a,i-1));

    V[a][b] = true;
    return T[a][b] = maxx;
}

int main() {
    while(cin >> n, n) {
        memset(S, 0, sizeof(S));
        memset(V, 0, sizeof(V));
        S[0] = 0;
        for(int i=1; i<=n; i++) {
            cin >> S[i];
            S[i] += S[i-1];
        }

        cout << 2*TT(1, n)-S[n]-S[0] << endl;

    }
}
```

uva/10930.cpp

```
//10930
//A-Sequence
//Dynamic Programming;Knapsack;Binary Knapsack
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

int K[30001];

int main() {
    int n, t=0, w;
    while(t++, cin >> n) {
        cout << "Case #" << t << ":";
        memset(K, 0, sizeof(K));

        bool ok=true;

        K[0] = 1; int last = 0;
        for(int i=1; i<=n; i++) {
            cin >> w; cout << " " << w;
            ok &= !K[w] && w > last;
            for(int j=10000; j>=w; j--)
                if (K[j-w])
                    K[j] = 1;
            last = w;
        }
        cout << endl;
```

```
        cout << "This is" << (ok?"":" not") << " an A-sequence." << endl;
    }

    return 0;
}
```

uva/10986.cpp

```
//10986
//Sending email
//Graphs;Shortest Path;Dijkstra
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 200010

using namespace std;

struct Edge {
    int v, c;
    Edge(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Edge& that) const { return c > that.c; }
};

vector<Edge> G[MAX];
priority_queue<Edge> Q;
int n, m, s, t;
int V[MAX];


int main() {
    int tt; cin >> tt; tt=0;
    while(cin >> n >> m >> s >> t) {
        int before = 0;
        memset(V, 0x3f, sizeof(V));
        memset(G, 0, sizeof(G));
        Q = priority_queue<Edge>();

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a].push_back(Edge(b, c));
            G[b].push_back(Edge(a, c));
            before += c;
        }

        int totalc=0;

        Q.push(Edge(s, 0));

        while(totalc < n && !Q.empty()) {
            Edge item = Q.top(); Q.pop();
            if (item.c >= V[item.v]) continue;

            V[item.v] = item.c;
            totalc++;

            for(int j=0; j<G[item.v].size(); j++) {
                Edge e = G[item.v][j];
                if (item.c + e.c < V[e.v])
                    Q.push(Edge(e.v, item.c + e.c));
            }
        }

        cout << "Case #" << ++tt << ": ";
        if (V[t] < 0x3f3f3f3f)
            cout << V[t] << endl;
        else
            cout << "unreachable" << endl;
    }
    return 0;
}
```

uva/11003.cpp

```
//11003
```

```cpp
//Boxes
//Dynamic Programming;Longest Increasing Subsequence
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#include <climits>
#define MAX 10005
using namespace std;

int T[MAX];
int main() {
    int n, w, c;
    while(cin >> n, n) {
        memset(T, 0, sizeof(T));

        int k = 0;
        T[0] = INT_MAX;
        for(int i=1; i<=n; i++) {
            cin >> w >> c;
            for(int j=k; j>=0; j--) {
                int next = min(T[j]-w, c);
                if (next >= T[j+1]) {
                    T[j+1] = next;
                    k=max(k, j+1);
                }
            }
        }

        cout << k << endl;
    }

    return 0;
}
```

uva/11059.cpp

```cpp
//11059
//Maximum Product
//Dynamic Programming;Maximum Sum Contiguous Subsequence
#include <iostream>
#include <climits>
#include <cmath>
#define MAX 1005
using namespace std;

int main() {
    long long n, a, t=0;
    while(cin >> n) {
        long long maxx=0, newneg=0, newpos=0, spos=1, sneg=1;
        bool valid = false;
        for(int i=0;i<n;i++) {
            cin >> a;
            if (spos*a>0) {
                valid = true;
                spos*=a;
            } else {
                newneg = spos*a;
                spos = 1;
            }


            if (sneg*a<0) {
                sneg*=a;
            } else  {
                if (sneg*a>0) valid = true;
                newpos = sneg*a;
                sneg = 1;
            }

            maxx = max(maxx, spos = max(spos, newpos));
            sneg = min(sneg, newneg);
            newpos = newneg = 0;
        }
        if (!valid) maxx = 0;
        cout << "Case #" << ++t << ": The maximum product is " << maxx << "." << endl;
        cout << endl;
    }

    return 0;
}
```

uva/11096.cpp

```cpp
//11096
//Nails
//Math;Geometry;Convex Hull;Monotone Chain
#include <iostream>
#include <cmath>
#include <iomanip>
#include <algorithm>
#define ull long long
using namespace std;

struct Point {
    ull x, y;

    ull product(Point a, Point b) {
        return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
    }

    bool right(Point a, Point b) {
        return product(a, b) > 0;
    }

    double dist(Point b) {
        return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y));
    }

    bool operator <(const Point& p) const {
        if (this->x != p.x) return this->x < p.x;
        return this->y < p.y;
    }

    bool operator ==(const Point& p) const {
        return this->x == p.x and this->y == p.y;
    }
};

int convexHull(Point* P, int n, Point* S) {
    sort(P, P+n);

    int m=0;
    for(int i=0; i<n; i++) {
        while(m >= 2 && !S[m-1].right(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    for(int i=n-1, k=m; i >= 0; i--) {
        while(m >= k+2 && !S[m-1].right(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    return m;
}


Point P[120], S[120];

int main() {
    int tt; cin >> tt;
    while(tt--) {
        int r, n;
        cin >> r >> n;
        for(int i=0; i<n; i++)
            cin >> P[i].x >> P[i].y;

        int s = convexHull(P, n, S);

        double final = 0.0;
        for(int i=0; i<s; i++)
            final += S[i].dist(S[(i+1)%n]);

        final = max(final, (double)r);

        cout << fixed << setprecision(5) << final << endl;
    }
}
```

uva/11110.cpp

```cpp
//11110
//Equidivisions
//Graphs;Flood Fill
#include <iostream>
#include <string>
#include <sstream>
#include <cstring>
#define MAX 102
using namespace std;

int G[MAX][MAX];
int n;

int fill(int x, int y, int v) {
    if (G[x][y] != v) return 0;
    if (x<=0 || x>n || y<=0 || y>n) return 0;

    G[x][y] = -1;
    return 1 +
        fill(x-1, y, v) + fill(x+1, y, v) +
        fill(x, y-1, v) + fill(x, y+1, v);
}

int main() {
    while(cin >> n, n) {
        int a, b; string s;
        memset(G, 0, sizeof(G));

        getline(cin, s);
        for(int i=1;i<n;i++) {
            getline(cin, s);
            stringstream sin(s);
            while(sin >> a >> b)
                G[a][b] = i;
        }
        bool good = true;
        for(int i=1;i<=n;i++) {
            for(int j=1;j<=n;j++) {
                if (G[i][j] >= 0)
                    good &= fill(i,j, G[i][j]) == n;
            }
        }

        cout << (good?"good":"wrong") << endl;

    }
    return 0;
}
```

uva/11151.cpp

```cpp
//11151
//Longest Palindrome
//Dynamic Programming;Longest Common Subsequence
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
string P, Q;

int main() {
    int p, q, t;
    cin >> t;
    getline(cin, P);
    while(t--) {
        getline(cin, P);
        Q = string(P.rbegin(), P.rend());
        int p = P.size(), q = Q.size();

        for(int i=0; i<=p; i++) { T[i][0] = 0;  }
        for(int i=0; i<=q; i++) { T[0][i] = 0;  }

        for(int i=1; i<=p; i++) {
            for(int j=1; j<=q; j++) {
                if (P[i-1] == Q[j-1]) {
                    T[i][j] = T[i-1][j-1] + 1;
                }
```

```
                else {
                    T[i][j] = max(T[i-1][j], T[i][j-1]);
                }
            }
        }

        cout << T[p][q] << endl;
    }

    return 0;
}
```

uva/11157.cpp

```
//11157
//Dynamic Frog
//Misc;Sort
#include <iostream>
#include <algorithm>
#include <vector>
using namespace std;

vector<int> V;

int main() {
    int t, n, d;
    cin >> t; t=0;

    while(cin >> n >> d) {
        char a; int b;
        V.clear();
        V.push_back(0);
        V.push_back(d);
        for(int i=0;i<n; i++) {
            cin >> a; cin.ignore(); cin >> b;
            V.push_back(b);
            if (a=='B')
                V.push_back(b);
        }
        sort(V.begin(), V.end());

        int maxx = 0;
        for(int i=3;i<V.size(); i+=2)
            maxx = max(maxx, V[i]-V[i-2]);

        for(int i=2;i<V.size(); i+=2)
            maxx = max(maxx, V[i]-V[i-2]);

        cout << "Case " << ++t << ": " << maxx << endl;
    }

}
```

uva/11159.cpp

```
//11159
//Factors and Multiples
//Graphs;Bipartite Matching
#include <iostream>
#include <cstring>
#include <climits>
#include <string>
#define MAX 205
using namespace std;

int A[MAX], B[MAX], G[MAX][MAX], n;
bool V[MAX];

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=0; i<=n; i++) {
        if (!V[i] && G[s][i] > 0) {
            if (int sent = send(i, t, min(minn, G[s][i]))) {
                G[s][i] -= sent;
                G[i][s] += sent;
                return sent;
            }
```

```
        }
    }
    return 0;
}

int main() {
    int t; cin >> t;

    for(int tt=1; tt<=t; tt++) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));

        int a; cin >> a;
        for(int i=1; i<=a; i++)
            cin >> A[i];

        int b; cin >> b;
        for(int i=1; i<=b; i++)
            cin >> B[i];

        for(int i=1; i<=a; i++) {
            G[0][i] = 1;
            for(int j=1; j<=b; j++)
                if (B[j]==0 || A[i] != 0 && B[j]%A[i] == 0)
                    G[i][a+j] = 1;
        }

        for(int i=a+1; i<=a+b; i++)
            G[i][a+b+1] = 1;

        n = a+b+1;

        int total = 0;
        while(int sent = send(0, n, INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }
        cout << "Case " << tt << ": " << total << endl;
    }

    return 0;
}
```

uva/11172.cpp

```
//579
//ClockHands
//Misc;Ad hoc
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

int main() {
    int n;
    cin >> n;
    while(n--) {
        int x, y;
        cin >> x >> y;
        if (x>y) {
            cout << ">" << endl;
        } else if (x<y) {
            cout << "<" << endl;
        } else {
            cout << "=" << endl;
        }
    }
}
```

uva/11235.cpp

```
//11235
//Frequent Values
//Misc;Segment Tree;Range Maximum Query
#include <iostream>
#include <cstring>
#include <cstdio>
#define MAX 600100
#define ull long long
```

```
 9  using namespace std;
10
11  struct Segtree {
12      int T[MAX], V[MAX];
13      int n;
14
15      Segtree() {
16          clear(1);
17      }
18
19      void clear(int n) {
20          this->n = n;
21          memset(V, 0, (n+1)*sizeof(int));
22          build(1, 1, n);
23      }
24
25      void build(int v, int a, int b) {
26          T[v] = a;
27
28          if (a>=b) return;
29          build(2*v, a, (a+b)/2);
30          build(2*v+1, (a+b)/2+1, b);
31      }
32
33      int maxv(int a, int b) {
34          return V[a]<V[b] ? b : a;
35      }
36
37      int query(int v, int a, int b, int i, int j) {
38          if (i>b || j<a || j<i)
39              return 0;
40
41          if (i<=a && b<=j)
42              return T[v];
43
44          return maxv(
45              query(v*2, a, (a+b)/2, i, j),
46              query(v*2+1, (a+b)/2+1, b, i, j));
47
48      }
49
50      int update(int v, int a, int b, int i, int x) {
51          if (a==b)
52              return V[a] = x, T[v] = a;
53          else if (i<=(a+b)/2)
54              return T[v] = maxv(T[v*2+1], update(v*2, a, (a+b)/2, i, x));
55          else
56              return T[v] = maxv(T[v*2], update(v*2+1, (a+b)/2+1, b, i, x));
57      }
58
59      int query(int i, int j) {
60          return query(1, 1, n, i, j);
61      }
62
63      int update(int i, int x) {
64          return update(1, 1, n, i, x);
65      }
66  };
67
68  Segtree T;
69  int S[MAX], E[MAX], C[MAX], N[MAX];
70
71  int main() {
72      int n, q;
73      while(cin >> n >> q, n) {
74          T.clear(n);
75
76          for(int i=1; i<=n; i++)
77              cin >> N[i];
78
79          for(int i=n; i>0; i--) {
80              if (i<n && N[i] == N[i+1])
81                  E[i] = E[i+1];
82              else
83                  E[i] = i;
84          }
85
86          for(int i=1; i<=n; i++) {
87              if (i>1 && N[i] == N[i-1])
88                  S[i] = S[i-1];
89              else
90                  S[i] = i;
```

```
            C[i] = E[i] - S[i] + 1;
            T.update(i, C[i]);
        }

        for(int i=0; i<q; i++) {
            int a, b; cin >> a >> b;
            if (N[a] == N[b])
                cout << b-a+1 << endl;
            else {
                int case1 = E[a] - max(S[a], a) + 1;
                int case2 = min(E[b], b) - S[b] + 1;
                int case3 = C[T.query(E[a] + 1, S[b]-1)];
                cout << max(case1, max(case2, case3)) << endl;
            }
        }
    }
}
```

uva/11280.cpp

```
//11280
//Flying to Fredericton
//Graphs;Shortest Path;Dijkstra
#include <iostream>
#include <cstring>
#include <string>
#include <vector>
#include <map>
#include <queue>
#include <algorithm>
#define MAX 1001
using namespace std;

struct Edge {
    int v, c, s;
    Edge(int v, int c, int s) : v(v), c(c), s(s) {}
    inline bool operator <(const Edge& a) const {
        return this->c > a.c;
    }
};

vector<Edge> G[MAX];
int V[MAX][MAX];
map<string, int> S;
int n, m;

int main() {
    int T; cin >> T;
    for(int tt=1; tt<=T; tt++) {
        S.clear();
        memset(G, 0, sizeof(G));
        memset(V, 0x7f, sizeof(V));

        cin >> n;
        for(int i=1; i<=n; i++) {
            string s; cin >> s;
            S[s] = i;
        }

        cin >> m;
        for(int i=1; i<=m; i++) {
            string s1, s2;
            int cost;
            cin >> s1 >> s2 >> cost;

            G[S[s1]].push_back(Edge(S[s2], cost, 0));
        }

        priority_queue<Edge> Q;
        Q.push(Edge(1, 0, 0));

        while(!Q.empty()) {
            Edge e = Q.top(); Q.pop();
            if (V[e.v][e.s] < e.c) continue;

            V[e.v][e.s] = e.c;

            for(int i=0; i<G[e.v].size(); i++) {
                Edge a = G[e.v][i];
                Q.push(Edge(a.v, e.c+a.c, e.s+1));
            }
```

```
        }

        if (tt>1) cout << endl;
        cout << "Scenario #" << tt << endl;
        int q; cin >> q;
        for(int i=0; i<q; i++) {
            int a; cin >> a;
            int minn = 0x7f7f7f7f;
            for(int j=0; j<=a+1; j++) {
                minn = min(minn, V[n][j]);
            }

            if (minn < 0x7f7f7f7f)
                cout << "Total cost of flight(s) is $" << minn << endl;
            else
                cout << "No satisfactory flights" << endl;
        }

    }
}
```

uva/11294.cpp

```
//11294
//Wedding
//Graphs;2-SAT
#include <iostream>
#include <string>
#include <cstring>
#include <vector>
#define MAX 1000
using namespace std;

vector<int> G[MAX*2], T[MAX*2];
int O[MAX*2], V[MAX*2], npv, n;
char R[MAX*2];

int neg(int x) {
    if (x>=n) return x-n;
    return x+n;
}

void set(int v, bool value) {
    if (R[v] != 0) return;
    R[v] = value ? 'w' : 'h';
    R[neg(v)] = value ? 'h': 'w';

    if (value)
        for(int i=0; i<G[v].size(); i++)
            set(G[v][i], true);
    else
        for(int i=0; i<G[neg(v)].size(); i++)
            set(G[neg(v)][i], true);
}

void DFS(int v){
    V[v] = 1;
    for(int i = 0; i < G[v].size(); i++)
        if (!V[G[v][i]])
            DFS(G[v][i]);
    O[npv++] = v;
}

void DFSt(int v, int comp){
    V[v] = comp;
    for(int i = 0; i < T[v].size(); i++)
       if (!V[T[v][i]])
            DFSt(T[v][i], comp);
}

int main() {
    int m;
    while(cin >> n >> m, n||m) {
        memset(G, 0, sizeof(G));
        memset(T, 0, sizeof(T));
        memset(R, 0, sizeof(R));

        for(int i=0;i<m; i++) {
            int a, b; char c, d;
            cin >> a >> c >> b >> d;
            if (c=='h') a=neg(a);
```

```
            if (d=='h') b=neg(b);

            G[neg(a)].push_back(b);
            G[neg(b)].push_back(a);
        }

        G[neg(0)].push_back(0);

        for(int i=0; i<2*n; i++)
            for(int j=0; j<G[i].size(); j++)
                T[G[i][j]].push_back(i);

        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i = 0; i < 2*n; i++)
            if(!V[i]) DFS(i);

        memset(V, 0, sizeof(V));

        int comp = 0;
        for(int i = 2*n-1; i >= 0; i--)
            if(!V[O[i]])
                DFSt(O[i], ++comp);

        bool result = true;
        for(int i=0; i<n; i++) {
            result &= V[i] != V[neg(i)];
        }

        if (!result) {
            cout << "bad luck" << endl;
            continue;
        }

        for(int i=1; i<=comp; i++) {
            for(int j=0; j<2*n; j++) {
                if (V[j] == i)
                    set(j, false);
            }
        }

        for(int i=1; i<n; i++) {
            if (i>1) cout << " ";
            cout << i << R[i];
        }
        cout << endl;
    }
}
```

uva/11297.cpp

```
//11297
//Census
//Misc;Segment Tree;2D
#include <iostream>
#include <cstring>
#include <functional>
#define MAX 506
#define ull long long
using namespace std;

int P[MAX][MAX];

struct Point {
    int x, y, mx;
    Point() : x(0), y(0), mx(-1) {}
    Point(int x, int y, int mx) : x(x), y(y), mx(mx) { }

    bool operator <(const Point& other) const {
        return mx < other.mx;
    }
};

struct Segtree2d {
    Point T[2*MAX*MAX];
    int n, m;

    void clear(int n, int m) {
        this->n = n;
```

```
        this->m = m;
        build(1, 1, 1, n, m);
    }

    int c(int s1, int s2) {
        return (s1+s2)/2;
    }

    Point build(int v, int a1, int b1, int a2, int b2) {
        if (a1>a2 || b1>b2) return def();

        if (a1 == a2 && b1 == b2)
            return T[v] = Point(a1, b1, P[a1][b1]);

        T[v] = def();
        T[v] = maxv(T[v], build(4*v-2, a1,          b1,          c(a1, a2), c(b1, b2)));
        T[v] = maxv(T[v], build(4*v-1, c(a1, a2)+1, b1,          a2,        c(b1, b2)));
        T[v] = maxv(T[v], build(4*v+0, a1,          c(b1, b2)+1, c(a1, a2), b2       ));
        T[v] = maxv(T[v], build(4*v+1, c(a1, a2)+1, c(b1, b2)+1, a2,        b2       ));
        return T[v];
    }

    //virtual apenas para permitir árvore de mínimo
    virtual Point maxv(Point a, Point b) {
        return max(a, b);
    }

    virtual Point def() {
        return Point(0, 0, -1);
    }


    Point query(int v, int a1, int b1, int a2, int b2, int x1, int y1, int x2, int y2) {
        if (x1>a2 || y1>b2 || x2<a1 || y2<b1 || a1>a2 || b1>b2)
            return def();

        if (x1<=a1 && y1<=b1 && a2<=x2 && b2<=y2)
            return T[v];

        Point mx = def();

        mx = maxv(mx, query(4*v-2, a1,          b1,          c(a1, a2), c(b1, b2), x1, y1, x2, y2));
        mx = maxv(mx, query(4*v-1, c(a1, a2)+1, b1,          a2,        c(b1, b2), x1, y1, x2, y2));
        mx = maxv(mx, query(4*v+0, a1,          c(b1, b2)+1, c(a1, a2), b2,        x1, y1, x2, y2));
        mx = maxv(mx, query(4*v+1, c(a1, a2)+1, c(b1, b2)+1, a2,        b2,        x1, y1, x2, y2));

        return mx;
    }

    Point query(int x1, int y1, int x2, int y2) {
        return query(1, 1, 1, n, m, x1, y1, x2, y2);
    }

    Point update(int v, int a1, int b1, int a2, int b2, int x, int y, int value) {
        if (a1>a2 || b1>b2) return def();

        if (x>a2 || y>b2 || x<a1 || y<b1)
            return T[v];

        if (x==a1 && y==b1 && x==a2 && y==b2)
            return T[v] = Point(x, y, value);

        Point mx = def();

        mx = maxv(mx, update(4*v-2, a1,          b1,          c(a1, a2), c(b1, b2), x, y, value));
        mx = maxv(mx, update(4*v-1, c(a1, a2)+1, b1,          a2,        c(b1, b2), x, y, value));
        mx = maxv(mx, update(4*v+0, a1,          c(b1, b2)+1, c(a1, a2), b2,        x, y, value));
        mx = maxv(mx, update(4*v+1, c(a1, a2)+1, c(b1, b2)+1, a2,        b2,        x, y, value));

        return T[v] = mx;
    }

    Point update(int x, int y, int value) {
        return update(1, 1, 1, n, m, x, y, value);
    }
};

struct Segtree2dMin : Segtree2d {
    Point maxv(Point a, Point b) {
        return min(a, b);
    }

```

```
    Point def() {
        return Point(0, 0, 1<<29);
    }
};

Segtree2d Tmax;
Segtree2dMin Tmin;

int main() {
    int n, m;
    while(cin >> n >> m) {
        for(int i=1; i<=n; i++)
            for(int j=1; j<=m; j++)
                cin >> P[i][j];

        Tmax.clear(n, m);
        Tmin.clear(n, m);


        int q; cin >> q;
        while(q--) {
            char cmd;
            cin >> cmd;

            if (cmd == 'q') {
                int x1, y1, x2, y2;
                cin >> x1 >> y1 >> x2 >> y2;
                cout << Tmax.query(x1, y1, x2, y2).mx << " " << Tmin.query(x1, y1, x2, y2).mx << endl;
            } else {
                int x, y, v;
                cin >> x >> y >> v;
                Tmax.update(x, y, v);
                Tmin.update(x, y, v);
            }
        }


    }
}
```

uva/11375.cpp

```
//11375
//Matches
//Dynamic Programming;Ad hoc
#include <iostream>
#include <vector>
#include <cstring>
using namespace std;

int K[] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6};
vector<int> T[2001][10];

void add(vector<int> &a, const vector<int> &b) {
    int carry = 0;
    for(int i=0;i<max(a.size(), b.size());i++) {
        int aa = i<a.size()?a[i]:0;
        int bb = i<b.size()?b[i]:0;
        int cc = aa+bb+carry;
        if (i >= a.size()) a.push_back(0);
        a[i] = cc%10;
        carry = cc/10;
    }
    if (carry)
        a.push_back(carry);
}

int main() {
    vector<int> one; one.push_back(1);

    for(int i=2; i<2001; i++) {
        for(int j=0;j<10; j++)
            if (i>=K[j]) {
                add(T[i][j], one);
                for(int k=0;k<10;k++)
                    add(T[i][j], T[i-K[j]][k]);
            }
    }

    int n;
    while(cin >> n) {
```

```
        vector<int> ans = n>=6?one:vector<int>();
        for(int i=1;i<10;i++)
            add(ans, T[n][i]);

        for(int i=ans.size()-1;i>=0;i--) {
            cout << ans[i];
        }
        if (ans.size()==0) cout << 0;
        cout << endl;
    }

    return 0;
}
```

uva/11402.cpp

```
//11402
//Ahoy, Pirates!
//Misc;Segment Tree;Lazy Propagation
#include <iostream>
#include <string>
#include <cstring>
#define MAX 3000100
#define ull long long
using namespace std;

struct Node {
    int a, b;
    int pending;

    Node() : a(0), b(0), pending(0) {}
    Node(int a) : a(a), b(0), pending(0) { }
    Node(int a, int b) : a(a), b(b), pending(0) { }

    Node change(int n) {
        if (n==1) {
            b += a;
            a = 0;
            pending = n;
        } else if (n==2) {
            a += b;
            b = 0;
            pending = n;
        } else if (n==3) {
            swap(a, b);
            pending = 3-pending;
        }

        return *this;
    }

    Node operator +(Node x) {
        return Node(a+x.a, b+x.b);
    }
};

struct Segtree {
    Node T[MAX];
    int n;

    void clear(int n, int *P) {
        this->n = n;

        build(1, 1, n, P);
    }

    Node build(int v, int a, int b, int *P) {
        if (a==b)
            return T[v] = Node(1-P[a], P[a]);
        else
            return T[v] =
                build(2*v, a, (a+b)/2, P) +
                build(2*v+1, (a+b)/2+1, b, P);
    }

    Node update(int v, int a, int b, int i, int j, int carry, int increment) {
        T[v].change(carry);

        if (i>b || j<a)
            return Node(0);

```

```
        if (i<=a && b<=j)
            return T[v].change(increment);

        Node answer =
            update(v*2, a, (a+b)/2, i, j, T[v].pending, increment) +
            update(v*2+1, (a+b)/2+1, b, i, j, T[v].pending, increment);

        T[v] = T[v*2] + T[v*2+1];

        return answer;
    }

    Node update(int i, int j, int inc) {
        return update(1, 1, n, i, j, 0, inc);
    }

    Node query(int i, int j) {
        return update(i, j, 0);
    }

};

Segtree T;
int P[MAX];
string s;

int main() {
    int cases; cin >> cases;

    for(int tt=1; tt<=cases; tt++) {
        cout << "Case " << tt << ":" << endl;

        int m; cin >> m;
        int n = 0;
        for(int i=0; i<m; i++) {
            int t;
            cin >> t >> s;
            for(int j=0; j<t; j++) {
                for(int k=0; k<s.size(); k++) {
                    P[++n] = s[k]-'0';
                }
            }
        }
        T.clear(n, P);

        int q; cin >> q;
        int query = 0;
        while(q--) {
            char cmd; int a, b;
            cin >> cmd >> a >> b;
            a++; b++;
            if (cmd == 'F') {
                T.update(a, b, 1);
            } else if (cmd == 'E') {
                T.update(a, b, 2);
            } else if (cmd == 'I') {
                T.update(a, b, 3);
            } else {
                Node node = T.query(a, b);
                cout << "Q" << ++query << ": " << node.b << endl;
            }
        }
    }
}
```

uva/11419.cpp

```
//11419
//SAM I AM
//Graphs;Bipartite Matching;Konig Theorem
#include <iostream>
#include <cstring>
#include <climits>
#include <string>
#include <cstdio>
#include <vector>
#define MAX 2005
using namespace std;

string VA[MAX], VB[MAX];
int G[MAX][MAX], n, r, c, p;
```

```
15  vector<int> G2[MAX];
16  bool V[MAX];
17
18  inline int SOURCE() { return 0; }
19  inline int TARGET() { return 1; }
20  inline int R(int i) { return 1+i; }
21  inline int C(int i) { return 1+r+i; }
22
23  int send(int s, int t, int minn) {
24      V[s] = true;
25
26      if (s==t) return minn;
27      for(int i=0; i<G2[s].size(); i++) {
28          int u = G2[s][i];
29          if (!V[u] && G[s][u] > 0) {
30              if (int sent = send(u, t, min(minn, G[s][u]))) {
31                  G[s][u] -= sent;
32                  G[u][s] += sent;
33                  return sent;
34              }
35          }
36      }
37      return 0;
38  }
39
40  void mark(int v, bool side) {
41      V[v] = true;
42      for(int i=0; i<G2[v].size(); i++) {
43          int u = G2[v][i];
44          if (!V[u] && (side && G[v][u] || !side && G[u][v]))
45              mark(i, !side);
46      }
47  }
48
49
50  int main() {
51      while(scanf("%d %d %d", &r, &c, &p), r|c|p) {
52          memset(G, 0, sizeof(G));
53          memset(G2, 0, sizeof(G2));
54          memset(V, 0, sizeof(V));
55
56          for(int i=1; i<=r; i++) {
57              G[SOURCE()][R(i)] = 1;
58              G2[SOURCE()].push_back(R(i));
59          }
60
61          for(int i=1; i<=c; i++) {
62              G[C(i)][TARGET()] = 1;
63              G2[C(i)].push_back(TARGET());
64          }
65
66
67          for(int i=0; i<p; i++) {
68              int a, b;
69              cin >> a >> b;
70              G[R(a)][C(b)] = 1;
71              G2[R(a)].push_back(C(b));
72              G2[C(b)].push_back(R(a));
73          }
74
75          n = r+c+1;
76
77          int total = 0;
78          while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
79              total += sent;
80              memset(V, 0, sizeof(V));
81          }
82
83          V[0] = V[1] = true;
84          for(int i=1; i<=r; i++) {
85              bool inflow = false;
86              for(int j=1; j<=c; j++)
87                  inflow |= G[C(j)][R(i)];
88
89              if (!V[R(i)] && !inflow)
90                  mark(R(i), true);
91          }
92          printf("%d", total);
93          for(int i=1; i<=r; i++)
94              if (!V[R(i)]) printf(" r%d", i);
95
96          for(int i=1; i<=c; i++)
```

```
            if (V[C(i)]) printf(" c%d", i);

        printf("\n");
    }

    return 0;
}
```

uva/11423.cpp

```
//11423
//Cache Simulator
//Misc;Fenwick Tree
#include <iostream>
#include <algorithm>
#include <cstring>
#define MAX 10000100
using namespace std;

struct Fenwick {
    int T[MAX];
    int n;

    Fenwick() {
        clear(MAX);
    }

    void clear(int n) {
        n++;
        memset(T, 0, n*sizeof(int));
        this->n = n;
    }

    void adjust(int k, int v) {
        for (; k < n; k += (k&-k))
            T[k] += v;
    }

    int rsq(int b) {
        int sum = 0;
        for (; b; b -= (b&-b))
            sum += T[b];
        return sum;
    }

    int rsq(int a, int b) {
        return rsq(b) - rsq(a - 1);
    }
};

Fenwick T;
int C[40], S[40], P[1<<24];
int caches=0, query=0;

void access(int addr) {
    if (P[addr]) {
        int maxCache = T.rsq(P[addr], query);

        int upto = lower_bound(C, C+caches, maxCache)-C;

        for(int i=0; i<upto; i++)
            S[i]++;

        T.adjust(P[addr], -1);
    } else {
        for(int i=0; i<caches; i++)
            S[i]++;
    }

    T.adjust(P[addr] = ++query, 1);
}

int main() {
    cin >> caches;
    for(int i=0; i<caches; i++)
        cin >> C[i];

    string cmd;
    while(cin >> cmd, cmd != "END") {
        if (cmd == "ADDR") {
            int x;
```

```
            cin >> x;
            access(x);
        }  else if (cmd == "RANGE") {
            int b, y, n;
            cin >> b >> y >> n;

            for(int i=0; i<n; i++)
                access(b+i*y);
        } else {
            for(int i=0; i<caches; i++) {
                if (i) cout << " ";
                cout << S[i];
            }
            cout << endl;
            memset(S, 0, sizeof S);
        }
    }



    return 0;
}
```

uva/11475.cpp

```
//11475
//Extend to Palindrome
//Misc;String Matching;KMP;Suffix-Prefix
#include <iostream>
#include <string>
#include <cstring>
#define MAX 100010
using namespace std;

int F[MAX];

void kmp_init(string& P) {
    F[0] = 0;  F[1] = 0;
    int i = 1, j = 0;
    while(i<P.size()) {
        if (P[i] == P[j])
            F[++i] = ++j;
        else if (j == 0)
            F[++i] = 0;
        else
            j = F[j];
    }
}

int kmp(string& P, string& T) {
    kmp_init(P);
    int i = 0, j = 0;
    int n = T.size(), m = P.size();

    while(i < n) {
        while(j < m) {
            if (P[j] == T[i]) {
                i++; j++;
            } else break;
        }
        if (j == 0) i++;
        if (i==n) return j;
        j = F[j];
    }
    return 0;
}


int main() {
    string S, P, T;
    while(cin >> S) {
        P = string(S.rbegin(), S.rend());

        string K = S.substr(0, S.size()-kmp(P, S));

        cout << S+string(K.rbegin(), K.rend()) << endl;
    }
}
```

uva/11494.cpp

```cpp
//11494
//Queen
//Misc;Ad hoc
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

int main() {
    int x, y, a, b;
    while(cin >> x >> y >> a >> b, x|y|a|b) {
        if (x==a && y==b)
            cout << 0 << endl;
        else if (x==a || y==b || x+y == a+b || x-y==a-b)
            cout << 1 << endl;
        else
            cout << 2 << endl;
    }
    return 0;
}
```

uva/11503.cpp

```cpp
//11503
//Virtual Friends
//Misc;Union-Find
#include <iostream>
#include <map>
#include <string>
#include <cstring>
#include <algorithm>
using namespace std;

int P[200001], C[200001];
map<string, int> M;

int parent(int v) {
    if (P[v] != v) {
        int p = P[v] = parent(P[v]);
        C[v] = C[p];
        return p;
    } else {
        return v;
    }
}

int person(string& s) {
    if (M.find(s) != M.end())
        return M[s];
    else {
        int r = M[s] = M.size();
        C[r] = 1; P[r] = r;
        return r;
    }
}

int main() {
    int t; cin >> t; t=0;
    int n;

    while(cin >> n) {
        M.clear();
        while(n--) {
            string p, q;
            cin >> p >> q;
            int a = person(p), b=person(q);
            int pa = parent(a), pb=parent(b);
            if (pa==pb) {
                cout << C[pa] << endl;
                continue;
            }
            if (pa < pb) swap(pa, pb);

            P[pb] = pa;
            cout << (C[pa]+=C[pb]) << endl;
        }

```

```
    }
}
```

uva/11512.cpp

```
//11512
//GATTACA
//Misc;String Matching;Suffix Array;Longest Common Prefix
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <cmath>
#define MAX 10050
using namespace std;

int RA[MAX], tempRA[MAX];
int SA[MAX], tempSA[MAX];
int C[MAX];
int Phi[MAX], PLCP[MAX], LCP[MAX];

void suffix_sort(int n, int k) {
    memset(C, 0, sizeof C);

    for (int i = 0; i < n; i++)
        C[i + k < n ? RA[i + k] : 0]++;

    int sum = 0;
    for (int i = 0; i < max(256, n); i++) {
        int t = C[i];
        C[i] = sum;
        sum += t;
    }

    for (int i = 0; i < n; i++)
        tempSA[C[SA[i] + k < n ? RA[SA[i] + k] : 0]++] = SA[i];

    memcpy(SA, tempSA, n*sizeof(int));
}

void suffix_array(string &s) {
    int n = s.size();

    for (int i = 0; i < n; i++)
        RA[i] = s[i] - 1;

    for (int i = 0; i < n; i++)
        SA[i] = i;

    for (int k = 1; k < n; k *= 2) {
        suffix_sort(n, k);
        suffix_sort(n, 0);

        int r = tempRA[SA[0]] = 0;
        for (int i = 1; i < n; i++) {
            int s1 = SA[i], s2 = SA[i-1];
            bool equal = true;
            equal &= RA[s1] == RA[s2];
            equal &= RA[s1+k] == RA[s2+k];

            tempRA[SA[i]] = equal ? r : ++r;
        }

        memcpy(RA, tempRA, n*sizeof(int));
    }
}

void lcp(string &s) {
    int n = s.size();

    Phi[SA[0]] = -1;
    for (int i = 1; i < n; i++)
        Phi[SA[i]] = SA[i-1];

    int L = 0;
    for (int i = 0; i < n; i++) {
        if (Phi[i] == -1) {
            PLCP[i] = 0;
            continue;
        }
        while (s[i + L] == s[Phi[i] + L])
```

```
            L++;

        PLCP[i] = L;
        L = max(L-1, 0);
    }

    for (int i = 1; i < n; i++)
        LCP[i] = PLCP[SA[i]];
}

int main() {
    int tt; cin >> tt;
    while(tt--) {
        string s; cin >> s;
        s += "\1";
        suffix_array(s);
        lcp(s);

        int maxx=0, start=0, count=0, last;
        for(int i=1; i<s.size(); i++) {
            if (LCP[i] > maxx) {
                maxx = LCP[i];
                start = i-1;
                count = 2;
            } else if (LCP[i] == maxx && start+count==i) {
                count++;
            }
        }

        if (maxx > 0)
            cout << s.substr(SA[start], maxx) << " " << count << endl;
        else
            cout << "No repetitions found!" << endl;
    }
}
```

uva/11518.cpp

```
//11518
//Dominos 2
//Graphs;Flood Fill
#include <iostream>
#include <vector>
#include <cstring>
#define MAX 10002
using namespace std;

vector<int> G[MAX];
bool V[MAX];
int n,m,l;

int dfs(int v) {
    if (V[v]) return 0;
    V[v] = true;
    int r = 1;
    for(int i=0;i<G[v].size(); i++)
        r+=dfs(G[v][i]);
    return r;
}

int main() {
    int t; cin >> t;
    while(cin >> n >> m >> l) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));

        for(int i=0;i<m;i++) {
            int a, b;
            cin >> a >> b;
            G[a].push_back(b);
        }
        int sum = 0;
        for(int i=0;i<l;i++) {
            int a;
            cin >> a;
            sum+=dfs(a);
        }

        cout << sum << endl;

    }
```

```
    return 0;
}
```

uva/11525.cpp

```
//11525
//Permutation
//Misc;Fenwick Tree
#include <iostream>
#include <cstring>
#define MAX 50100
using namespace std;

struct Fenwick {
    int T[MAX];
    int n;

    Fenwick() {
        clear(0);
    }

    void clear(int n) {
        n++;
        memset(T, 0, n*sizeof(int));
        this->n = n;
    }

    void adjust(int k, int v) {
        for (; k < n; k += (k&-k))
            T[k] += v;
    }

    int rsq(int b) {
        int sum = 0;
        for (; b; b -= (b&-b))
            sum += T[b];
        return sum;
    }

    int rsq(int a, int b) {
        return rsq(b) - rsq(a - 1);
    }

    int lower_bound(int x) {
        int first = 0, count = n;
        while (count>0)
        {
            int step=count/2;
            int mid = first+step;

            if (rsq(mid) < x) {
                first = mid+1;
                count -= step+1;
            } else {
                count = step;
            }
        }
        return first;
    }
};

Fenwick T;

int main() {
    int tt; cin >> tt;
    while(tt--) {
        int n; cin >> n;

        T.clear(n);
        for(int i=1; i<=n; i++)
            T.adjust(i, 1);


        for(int i=0; i<n; i++) {
            int k; cin >> k;

            int x = T.lower_bound(k+1);
            T.adjust(x, -1);

            if (i) cout << " ";
            cout << x;
```

```
        }
        cout << endl;
    }

    return 0;
}
```

uva/11532.cpp

```
//11532
//Simple Adjacency Maximization
//Misc;Binary Manipulation
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

long long T[51][51];

int main() {
    for(int i=1;i<=50;i++) {
        for(int j=0;i+j<=50;j++) {
            int p=i, q=j;
            long long n = 0L;
            if (p%2!=0 && p/2<q) {
                n = 1;
                p--;
            }

            for(;p>1;p-=2) {
                if (q>0)
                    n = (n<<3) | 5L;
                else
                    n = (n<<2) | 3L;
                q--;
            }

            if (p==1) n = (n<<1) | 1L;
            T[i][j] = n;
        }
    }


    int t, p, q;
    cin >> t;
    while(cin >> p >> q) {
        cout << T[p][q] << endl;
    }

    return 0;
}
```

uva/11576.cpp

```
//11576
//Scrolling Sign
//Misc;String Matching;KMP;Suffix-Prefix
#include <iostream>
#include <string>
#include <cstring>
#define MAX 100010
using namespace std;

int F[MAX];

void kmp_init(string& P) {
    F[0] = 0;  F[1] = 0;
    int i = 1, j = 0;
    while(i<P.size()) {
        if (P[i] == P[j])
            F[++i] = ++j;
        else if (j == 0)
            F[++i] = 0;
        else
            j = F[j];
    }
}

int kmp(string& P, string& T) {
```

```
    kmp_init(P);
    int i = 0, j = 0;
    int n = T.size(), m = P.size();

    while(i < n) {
        while(j < m) {
            if (P[j] == T[i]) {
                i++; j++;
            } else break;
        }
        if (j == 0) i++;
        if (i==n) return j;
        j = F[j];
    }
    return 0;
}


int main() {
    int t; cin >> t; t=0;
    int k, w;
    while(cin >> k >> w) {
        int sum = 0;
        string Q, P = "";
        while(w--) {
            cin >> Q;
            sum += k-kmp(Q, P);
            P = Q;
        }
        cout << sum << endl;
    }
}
```

uva/11590.cpp

```
//11590
//Prefix Lookup
//Misc;String Matching;Trie
#include <iostream>
#include <cstring>
#define MAXS 1500010
#define ull unsigned long long
using namespace std;

struct Trie {
    int G[MAXS][2];
    ull S[MAXS];
    bool E[MAXS];
    int stateCount;

    Trie() {
        clear();
    }

    void clear() {
        stateCount = 0;
        clear(stateCount++);
    }

    int clear(int state) {
        memset(G[state], -1, sizeof G[state]);
        S[state] = 0;
        E[state] = false;
        return state;
    }

    void add(string &s) {
        int state = 0;
        for(int i=0; i<s.size()-1; i++) {
            S[state]++;

            int next = s[i] - '0';

            if (G[state][next] < 0)
                G[state][next] = clear(stateCount++);

            state = G[state][next];
        }
        E[state] = true;
    }
};
```

```
47
48  Trie T;
49
50  ull dfs(int state, int depth) {
51      ull s = 0;
52
53      for (int e = 0; e < 2; ++e) {
54          if (T.G[state][e] == -1) continue;
55
56          s += dfs(T.G[state][e], depth-1);
57      }
58
59      T.S[state] = s;
60
61      return T.E[state] ? 1ull << depth : s;
62  }
63
64
65  ull answer(string &s, int m) {
66      int state = 0;
67      for(int i=0; i<s.size()-1; i++) {
68          int next = s[i]-'0';
69          state = T.G[state][next];
70      }
71      int shift = m-s.size()+1;
72      ull base = shift == 64 ? 0 : 1ull << shift;
73      return base - T.S[state];
74  }
75
76  int main() {
77      int n, m, q;
78      while(cin >> n >> m, n|m) {
79          T.clear();
80
81          for(int i=0; i<n; i++) {
82              string s; cin >> s;
83              T.add(s);
84          }
85
86          dfs(0, m);
87
88          cin >> q;
89          for(int i=0; i<q; i++) {
90              string s; cin >> s;
91              cout << answer(s, m) << endl;
92          }
93          cout << endl;
94      }
95  }
```

uva/11597.cpp

```
1   //11597
2   //Spanning Subtree
3   //Misc;Ad hoc
4   #include <iostream>
5   using namespace std;
6
7   int main() {
8       int n, t=0;
9       while(cin >> n, t++, n) {
10          cout << "Case " << t << ": " << n/2 << endl;
11      }
12
13      return 0;
14  }
```

uva/11610.cpp

```
1   //11610
2   //Reverse Prime
3   //Misc;Fenwick Tree
4   #include <iostream>
5   #include <algorithm>
6   #include <cstring>
7   #define MAX 1000100
8   using namespace std;
9
10  struct Fenwick {
```

```
    int T[MAX];
    int n;

    Fenwick() {
        clear(MAX);
    }

    void clear(int n) {
        n++;
        memset(T, 0, n*sizeof(int));
        this->n = n;
    }

    void adjust(int k, int v) {
        for (; k < n; k += (k&-k))
            T[k] += v;
    }

    int rsq(int b) {
        int sum = 0;
        for (; b; b -= (b&-b))
            sum += T[b];
        return sum;
    }

    int rsq(int a, int b) {
        return rsq(b) - rsq(a - 1);
    }

    int value(int b) {
        return rsq(b, b);
    }

    int lower_bound(int x) {
        int first = 0, count = n;
        while (count>0)
        {
            int step=count/2;
            int mid = first+step;

            if (rsq(mid) < x) {
                first = mid+1;
                count -= step+1;
            } else {
                count = step;
            }
        }
        return first;
    }
};

Fenwick T, Q;
int P[MAX], W[MAX], wn=0, I[MAX], in=0, F[MAX];

int invert(int n) {
    int r=0;
    while(n) {
        r *= 10;
        r += n%10;
        n/=10;
    }
    return r;
}

int factors(int n, int start) {
    if (F[n]) return F[n];
    if (not P[n]) return F[n] = 1;

    for(;;start++)
        if (n%W[start] == 0)
            return factors(n/W[start], start)+1;
}


int main() {
    P[1] = true;
    for(long long i=2; i<MAX; i++) {
        if (P[i]) continue;
        W[wn++] = i;
        for(long long j=i*i; j<MAX; j+=i)
            P[j] = true;
    }
```

```
93
94
95      for(int i=100000; i<1000000; i++) {
96          if (not P[invert(i)]) {
97              I[i]=++in;
98              Q.adjust(in, 1);
99              T.adjust(in, factors(i, 0)+2);
100         }
101     }
102
103     char c; int n;
104     while(cin >> c >> n) {
105         if (c=='q') {
106             cout << T.rsq(Q.lower_bound(n+1)) << endl;
107         } else {
108             n/=10;
109             T.adjust(I[n], -T.value(I[n]));
110             Q.adjust(I[n], -1);
111         }
112     }
113
114
115     return 0;
116 }
```

uva/11626.cpp

```
1   //11626
2   //Convex Hull
3   //Math;Geometry;Point Sort
4   #include <iostream>
5   #include <algorithm>
6   #define long2 long long
7   using namespace std;
8
9   struct Point {
10      long2 x, y;
11
12      Point() {}
13      Point(long2 x, long2 y) : x(x), y(y) {}
14
15      long2 signal(Point& a, Point& b) {
16          return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
17      }
18
19      inline bool operator <(const Point& p) const {
20          if (this->x != p.x) return this->x < p.x;
21          return this->y < p.y;
22      }
23  };
24
25  Point P[100050], L[100050], U[100050], C[100050];
26
27  void print(Point* P, int n) {
28      for(int i=0; i<n; i++)
29          cout << P[i].x << " " << P[i].y << endl;
30  }
31
32  void printInv(Point* P, int n) {
33      for(int i=n-1; i>=0; i--)
34          cout << P[i].x << " " << P[i].y << endl;
35  }
36
37  int main() {
38      int t; cin >> t;
39      while(t--) {
40          int n;
41          cin >> n;
42          int m = 0;
43          for(int i=0; i<n; i++) {
44              long2 x, y; char c;
45              cin >> x >> y >> c;
46              if (c=='Y')
47                  P[m++] = Point(x,y);
48          }
49          sort(P, P+m);
50          int up=0, lo=0, ce=0;
51          for(int i=1; i<m-1; i++) {
52              long2 signal = P[i].signal(P[0], P[m-1]);
53              if (signal < 0)
54                  U[up++] = P[i];
```

```
            else if (signal > 0)
                L[lo++] = P[i];
            else
                C[ce++] = P[i];
        }


        cout << m << endl;
        cout << P[0].x << " " << P[0].y << endl;

        if (lo > 0)
            print(L, lo);
        else
            print(C, ce);

        cout << P[m-1].x << " " << P[m-1].y << endl;

        if (up > 0)
            printInv(U, up);
        else
            printInv(C, ce);
    }

}
```

uva/11629.cpp

```
//11629
//Ballot evaluation
//Misc;STL map
#include <iostream>
#include <map>
#include <cstring>
using namespace std;

map<string, int> P;

int main() {
    int n, g;
    while(cin >> n >> g) {
        P.clear();
        for(int i=0; i<n; i++) {
            string s; int a, b;
            cin >> s >> a; cin.get(); cin >> b;
            P[s] = a*10+b;
        }

        for(int i=1; i<=g; i++) {
            string s = "+"; int d=0; int r;

            while(s=="+") {
                cin >> s;
                d += P[s];
                cin >> s;
            }
            cin >> r; r*=10;

            bool result;
            if ( s=="<") result = d < r;
            if ( s=="<=") result = d <= r;
            if ( s==">") result = d > r;
            if ( s==">=") result = d >= r;
            if ( s=="=") result = d == r;
            cout << "Guess #" << i << " was " << (result?"correct":"incorrect") << "." << endl;
        }

    }
}
```

uva/11631.cpp

```
//11631
//Dark roads
//Graphs;Minimum Spanning Tree;Prim;Priority Queue
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
```

```cpp
#include <queue>
#define MAX 200010

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};

vector<Road> G[MAX];
priority_queue<Road> Q;
int n, m;
bool V[MAX];


int main() {
    while(cin >> n >> m, n|m) {
        int before = 0;
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        Q = priority_queue<Road>();

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
            before += c;
        }

        int total = 0, totalc=0;

        Q.push(Road(0, 0));

        while(totalc < n) {
            Road item = Q.top(); Q.pop();
            if (V[item.v]) continue;

            V[item.v] = true;
            total += item.c;
            totalc++;

            for(int j=0; j<G[item.v].size(); j++)
                if (!V[G[item.v][j].v])
                    Q.push(G[item.v][j]);
        }

        cout << before-total << endl;
    }
    return 0;
}
```

uva/11658.cpp

```cpp
//11658
//Best Coalitions
//Dynamic Programming;Knapsack;Binary Knapsack
#include <iostream>
#include <cstring>
#include <iomanip>
using namespace std;

int K[10001], W[102];

int main() {
  int n, x, a, b;

  while(cin >> n >> x, n|x) {
    memset(K, 0, sizeof(K));

    for(int i=1; i<=n; i++) {
        cin >> a; cin.ignore(); cin >> b;
        W[i] = a*100+b;
    }

    K[W[x]] = 1;
    for(int i=1; i<=n; i++) {
        if (i==x) continue;
        for(int j=10000; j>=W[i]; j--)
```

```
            if (K[j-W[i]])
                K[j] = 1;
    }

    int maxx = 0;
    for(int i=5001; i<=10000; i++) {
        if (K[i]) {
            maxx = i;
            break;
        }
    }

    cout << fixed << setprecision(2) << (W[x]/((double)maxx)*100.0) << endl;
  }

  return 0;
}
```

uva/11686.cpp

```
//11686
//Pick up Sticks
//Graphs;Topological Sorting
#include <iostream>
#include <cstdio>
#include <vector>
#include <cstring>
#define MAX 1000001
using namespace std;

int V[MAX];
int O[MAX], npv;
vector<int> G[MAX];
int n, m;

bool DFS(int d, int v){
    V[v] = 1;

    for(int i=0;i<G[v].size(); i++) {
        int u = G[v][i];
        if (V[u] == 1) return false;
        if (!V[u] && !DFS(d, u)) return false;
    }
    O[++npv] = v;
    V[v] = 2;
    return true;
}


int main() {
    int a, b;
    while(scanf("%d%d",&n, &m), n|m) {
        for(int i=1;i<=n;i++) G[i].clear();
        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        while(m--) {
            scanf("%d%d",&a, &b);
            G[a].push_back(b);
        }


        bool ok = true;
        int d = 0;
        for(int i = 1; i <= n; i++)
            if (!V[i])
                ok &= DFS(++d, i);

        if (ok)
            for(int i = n; i > 0; i--)
                printf("%d\n", O[i]);
        else
            printf("IMPOSSIBLE\n");
    }

    return 0;
}
```

uva/11703.cpp

```cpp
//11703
//sqrt log sin
//Dynamic Programming;Ad hoc
#include <iostream>
#include <cmath>
#include <cstring>
#include <cassert>
using namespace std;

int K[1000001];

int main() {
    K[0] = 1;
    for(int i=1; i<1000001; i++) {
        int a = (int)(i-sqrt(i));
        int b = (int)log(i);
        int c = (int)(i*pow(sin(i), 2));
        K[i] = (K[a] + K[b] + K[c])%1000000;
    }

    int n;
    while(cin >> n, n>-1)
        cout << K[n] << endl;

    return 0;
}
```

uva/11709.cpp

```cpp
//11709
//Trust Groups
//Graphs;Strongly Connected Components
#include <iostream>
#include <map>
#include <string>
#include <cstring>
#define MAX 1001
using namespace std;

map<string, int> P;
int person(const string& p) {
    if (P.find(p) != P.end())
        return P[p];
    else
        return P[p] = P.size();
}

bool V[MAX];
int O[MAX], npv;
bool G[MAX][MAX];
int n, m;

void DFS(int v){
    V[v] = true;
    for(int i = 1; i <= n; i++)
        if (G[v][i] && !V[i])
            DFS(i);
    O[++npv] = v;
}

void DFSt(int v){
    V[v] = true;
    for(int i = 1; i <= n; i++)
       if (G[i][v] && !V[i])
            DFSt(i);
}


int main() {
    int a, b, t; string p, q;
    while(cin >> n >> m, n|m) {
        memset(G, 0, sizeof(G));
        P.clear();
        getline(cin, p);

        for(int i=0; i<n; i++) getline(cin, p);

```

```
        while(m--) {
            getline(cin, p);
            getline(cin, q);
            G[person(p)][person(q)] = true;
        }

        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i = 1; i <= n; i++)
            if(!V[i]) DFS(i);

        memset(V, 0, sizeof(V));

        int comp = 0;
        for(int i = n; i > 0; i--)
            if(!V[O[i]]) {
                comp++;
                DFSt(O[i]);
            }

        cout << comp << endl;
    }

    return 0;
}
```

uva/11733.cpp

```
//11733
//Airports
//Graphs;Minimum Spanning Tree;Prim;Priority Queue
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 10005

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};

vector<Road> G[MAX];
int CStart[MAX], CCount[MAX], nc;
priority_queue<Road> Q;
int n, m, cca;
bool V[MAX];

int dfs(int v) {
    V[v] = true;
    int acum = 1;
    for(int i=0; i<G[v].size(); i++)
        if (!V[G[v][i].v])
            acum += dfs(G[v][i].v);
    return acum;
}

int main() {
    int t; cin >> t; t=0;
    while(cin >> n >> m >> cca) {
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        nc = 0;

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            if (c<cca) {
                G[a].push_back(Road(b, c));
                G[b].push_back(Road(a, c));
            }
        }

        for(int i=1; i<=n; i++) {
```

```
            if (!V[i]) {
                CStart[nc]=i;
                CCount[nc]=dfs(i);
                nc++;
            }
        }

        int total = nc*cca;

        for(int i=0; i<nc; i++) {
            int totalc = 0;
            Q = priority_queue<Road>();
            Q.push(Road(CStart[i], 0));
            memset(V, 0, sizeof(V));

            while(totalc < CCount[i]) {
                Road item = Q.top(); Q.pop();
                if (V[item.v]) continue;

                V[item.v] = true;
                total += item.c;
                totalc++;

                for(int j=0; j<G[item.v].size(); j++)
                    if (!V[G[item.v][j].v])
                        Q.push(G[item.v][j]);
            }
        }

        cout << "Case #" << ++t << ": " << total << " " << nc << endl;
    }
    return 0;
}
```

uva/11747.cpp

```
//11747
//Heavy Cycle Edges
//Graphs;Minimum Spanning Tree;Prim;Priority Queue
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 10005

using namespace std;

struct Road {
    int v, c;
    Road(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Road& that) const { return c > that.c; }
};

vector<Road> G[MAX];
int CStart[MAX], CCount[MAX], nc;
priority_queue<Road> Q;
vector<int> R;
int n, m;
bool V[MAX];

int dfs(int v) {
    V[v] = true;
    int acum = 1;
    for(int i=0; i<G[v].size(); i++)
        if (!V[G[v][i].v])
            acum += dfs(G[v][i].v);
    return acum;
}

int main() {
    while(cin >> n >> m, n|m) {
        memset(V, 0, sizeof(V));
        memset(G, 0, sizeof(G));
        nc = 0;
        R.clear();

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
```

```
            G[a].push_back(Road(b, c));
            G[b].push_back(Road(a, c));
        }

        for(int i=1; i<=n; i++) {
            if (!V[i]) {
                CStart[nc]=i;
                CCount[nc]=dfs(i);
                nc++;
            }
        }

        for(int i=0; i<nc; i++) {
            int totalc=0;
            Q.push(Road(CStart[i], 0));
            memset(V, 0, sizeof(V));

            while(totalc < CCount[i]) {
                Road item = Q.top(); Q.pop();
                if (V[item.v]) { R.push_back(item.c); continue; }

                V[item.v] = true;
                totalc++;

                for(int j=0; j<G[item.v].size(); j++)
                    if (!V[G[item.v][j].v])
                        Q.push(G[item.v][j]);
            }
            while(!Q.empty()) {
                R.push_back(Q.top().c);
                Q.pop();
            }
        }
        sort(R.begin(), R.end());
        if (R.size()==0) {
            cout << "forest" << endl;
        } else {
            cout << R[0];
            for(int i=1; i<R.size(); i++)
                cout << " " << R[i];
            cout << endl;
        }
    }
    return 0;
}
```

uva/11762.cpp

```
//11762
//Race to 1
//Math;Probability
#include <iostream>
#include <iomanip>
#define MAX 1000010
using namespace std;

int P[MAX], K[MAX], D[MAX][30];
double R[MAX];

int main() {
    for(int i=2; i<MAX; i++) {
        K[i] = K[i-1];

        if (P[i] > 0) continue;
        K[i]++;

        for(int j=i, k=1; j<MAX; j+=i, k++)
            D[j][P[j]++] = k;
    }

    R[1] = 0;
    for(int i=2; i<MAX; i++) {
        for(int j=0; j<P[i]; j++)
            R[i] += R[D[i][j]];

        R[i] /= P[i];
        R[i] += (double)K[i] / P[i];
    }

    int t; cin >> t;
    for(int tt = 1; tt<=t; tt++) {
```

```
        int a; cin >> a;
        cout << "Case " << tt << ": " << fixed << setprecision(10) << R[a] << endl;
    }
}
```

uva/11770.cpp

```
//11770
//Lighting Away
//Graphs;Topological Sorting
#include <iostream>
#include <vector>
#include <cstring>
#define MAX 10001
using namespace std;

bool V[MAX];
int O[MAX], npv;
vector<int> G[MAX];
int n, m;

void DFS(int v){
    if (V[v]) return;
    V[v] = true;
    for(int i = 0; i < G[v].size(); i++)
        DFS(G[v][i]);
    O[++npv] = v;
}

void DFSt(int v){
    if (V[v]) return;
    V[v] = true;
    for(int i = 0; i < G[v].size(); i++)
        DFSt(G[v][i]);
}


int main() {
    int a, b;
    int t; cin >> t; t=0;
    while(cin >> n >> m) {
        memset(G, 0, sizeof(G));

        while(m--) {
            cin >> a >> b;
            G[a].push_back(b);
        }

        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i = 1; i <= n; i++)
            if(!V[i]) DFS(i);

        memset(V, 0, sizeof(V));

        int comp = 0;
        for(int i = n; i > 0; i--)
            if(!V[O[i]]) {
                comp++;
                DFSt(O[i]);
            }

        cout << "Case " << ++t << ": " <<  comp << endl;
    }

    return 0;
}
```

uva/11833.cpp

```
//11833
//Route Change
//Graphs;Shortest Path;Dijkstra
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
```

```cpp
#include <algorithm>
#include <queue>
#define MAX 252

using namespace std;

struct Edge {
    int v, c;
    Edge(int v, int c) : v(v), c(c) {}
    inline bool operator < (const Edge& that) const { return c > that.c; }
};

int G[MAX][MAX];
int V[MAX], S[MAX];
int n, m, cc, kk;

int main() {
    while(cin >> n >> m >> cc >> kk, n|m|cc|kk) {
        memset(V, 0x3f, sizeof(V));
        memset(S, 0, sizeof(S));
        memset(G, -1, sizeof(G));

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a][b] = G[b][a] = c;
        }

        for(int i=cc-2; i>=0; i--) {
            S[i] = S[i+1] + G[i][i+1];
        }

        int totalc=0;

        priority_queue<Edge> Q;
        Q.push(Edge(kk, 0));

        while(totalc < n && !Q.empty()) {
            Edge item = Q.top(); Q.pop();
            if (item.c >= V[item.v]) continue;
            V[item.v] = item.c;
            totalc++;
            if (item.v < cc) continue;
            for(int j=0; j<n; j++) {
                if (G[item.v][j]>=0) {
                    Edge e = Edge(j, G[item.v][j]);
                    if (item.c + e.c < V[e.v])
                        Q.push(Edge(e.v, item.c + e.c));
                }
            }
        }

        int minn = 0x3f3f3f3f;
        for(int i=0;i<cc;i++) {
            minn = min(minn, V[i]+S[i]);
        }
        cout << minn << endl;
    }
    return 0;
}
```

uva/11838.cpp

```cpp
//11838
//Come and Go
//Graphs;Strongly Connected Components
#include <iostream>
#include <cstring>
#define MAX 1001
using namespace std;

bool V[MAX];
int O[MAX], npv;
bool G[MAX][MAX];
int n, m;

void DFS(int v){
    V[v] = true;
    for(int i = 1; i <= n; i++)
        if (G[v][i] && !V[i])
            DFS(i);
```

```
    O[++npv] = v;
}

void DFSt(int v){
    V[v] = true;
    for(int i = 1; i <= n; i++)
       if (G[i][v] && !V[i])
            DFSt(i);
}


int main() {
    int a, b, t;
    while(cin >> n >> m, n|m) {
        memset(G, 0, sizeof(G));

        while(m--) {
            cin >> a >> b >> t;
            G[a][b] = true;
            if (t==2)
                G[b][a] = true;
        }

        npv = 0;
        memset(V, 0, sizeof(V));
        memset(O, 0, sizeof(O));

        for(int i = 1; i <= n; i++)
            if(!V[i]) DFS(i);

        memset(V, 0, sizeof(V));

        int comp = 0;
        for(int i = n; i > 0; i--)
            if(!V[O[i]]) {
                comp++;
                DFSt(O[i]);
            }

        cout << (comp==1) << endl;
    }

    return 0;
}
```

uva/11857.cpp

```
//11857
//Driving Range
//Graphs;Minimum Spanning Tree;Kruskal
#include <iostream>
#include <cstring>
#include <vector>
#include <algorithm>
#include <cassert>
using namespace std;

struct Edge {
    int x, y, v;
    inline bool operator <(const Edge& that) const {
        return this->v < that.v;
    }
};

Edge E[1000005];
int P[1000005];

inline int findset(int v) {
    if (P[v] != v)
        return P[v] = findset(P[v]);
    return v;
}

inline int unionset(int x, int y) {
    int a = findset(x), b = findset(y);
    if (a==b) return -1;
    if (a>b) swap(a,b);
    P[b] = a;
    return a;
}

```

```
int main() {
    int n, m;
    while(cin >> n >> m, n||m) {
        for(int i=0; i<n; i++)
            P[i] = i;

        for(int i=0; i<m; i++)
            cin >> E[i].x >> E[i].y >> E[i].v;

        sort(E, E+m);

        int maxx=0, count=0;
        for(int i=0; i<m && count < n-1; i++) {
            if(unionset(E[i].x, E[i].y) != -1) {
                maxx = max(maxx, E[i].v);
                count++;
            }
        }
        if (count == n-1)
            cout << maxx << endl;
        else
            cout << "IMPOSSIBLE" << endl;
    }
}
```

uva/11966.cpp

```
//11966
//Galactic Bonding
//Misc;Union-Find
#include <iostream>
#include <map>
#include <string>
#include <cstring>
#include <algorithm>
#include <cmath>
using namespace std;

int P[1000];
double X[1000], Y[1000];

inline int findset(int v) {
    if (P[v] == v) return v;
    return P[v] = findset(P[v]);
}

inline bool unionset(int x, int y) {
    int a = findset(x), b = findset(y);
    if (a==b) return false;
    P[b] = a;
    return true;
}

inline double dist(int a, int b) {
    return pow(X[a]-X[b], 2.0)+pow(Y[a]-Y[b], 2.0);
}

int main() {
    int t; cin >> t; t=0;

    int n; double d;
    while(cin >> n >> d) {
        for(int i=0; i<n; i++) P[i] = i;

        int sets = n;
        for(int i=0; i<n; i++) {
            cin >> X[i] >> Y[i];
            for(int j=0;j<i;j++)
                if (dist(i,j)<=d*d && unionset(i, j))
                    sets--;
        }

        cout << "Case " << ++t << ": " << sets << endl;
    }
}
```

uva/12086.cpp

```
//12086
```

```cpp
//Potentiometers
//Misc;Fenwick Tree
#include <iostream>
#include <cstring>
#include <string>
#define MAX 200100
using namespace std;

struct Fenwick {
    int T[MAX];
    int n;

    Fenwick() {
        clear(0);
    }

    void clear(int n) {
        n++;
        memset(T, 0, n*sizeof(int));
        this->n = n;
    }

    void adjust(int k, int v) {
        for (; k < n; k += (k&-k))
            T[k] += v;
    }

    void update(int k, int v) {
        adjust(k, v-rsq(k, k));
    }

    int rsq(int b) {
        int sum = 0;
        for (; b; b -= (b&-b))
            sum += T[b];
        return sum;
    }

    int rsq(int a, int b) {
        return rsq(b) - rsq(a - 1);
    }

};

Fenwick T;

int main() {
    int n, tt=0;
    while(cin >> n, n) {
        if (tt++) cout << endl;
        cout << "Case " << tt << ":" << endl;

        T.clear(n);
        for(int i=1; i<=n; i++) {
            int a; cin >> a;
            T.adjust(i, a);
        }

        string cmd;
        while(cin >> cmd, cmd!="END") {
            int a, b; cin >> a >> b;
            if (cmd == "S") {
                T.update(a, b);
            } else {
                cout << T.rsq(a, b) << endl;
            }
        }
    }
}
```

uva/12101.cpp

```cpp
//12101
//Prime Path
//Graphs;Shortest Path;BFS
#include <iostream>
#include <queue>
#include <cstring>
#include <string>
using namespace std;

```

```
bool P[10000], V[10000];

struct Step {
    int a, b, c, d, w;
    Step() {}
    Step(int a, int b, int c, int d, int w) : a(a), b(b), c(c), d(d), w(w) {}

    int number() { return a*1000+b*100+c*10+d; }
    bool valid() { return a && P[number()]; }

    Step atA(int n) { return Step(n, b, c, d, w+1); }
    Step atB(int n) { return Step(a, n, c, d, w+1); }
    Step atC(int n) { return Step(a, b, n, d, w+1); }
    Step atD(int n) { return Step(a, b, c, n, w+1); }
};

Step makestep(int n) {
    int a, b, c, d;
    d = n%10; n/=10;
    c = n%10; n/=10;
    b = n%10; n/=10;
    a = n%10; n/=10;
    return Step(a,b,c,d,0);
}

int main() {
    memset(P, true, sizeof(P));
    P[0] = P[1] = false;
    for(int i=2; i<10000; i++) {
        if (P[i]) {
            for(int j=i*i; j<10000; j+=i)
                P[j] = false;
        }
    }
    int t, a, b;
    cin >> t;
    while(cin >> a >> b) {
        memset(V, 0, sizeof(V));
        queue<Step> Q;
        Q.push(makestep(a));
        bool found = false;
        while(!Q.empty()) {
            Step step = Q.front(); Q.pop();
            int n = step.number();
            if (V[n]) continue;
            V[n] = true;
            if (n == b) {
                cout << step.w << endl;
                found = true;
                break;
            }
            for(int i=0;i<=9;i++) {
                Step sa = step.atA(i);
                Step sb = step.atB(i);
                Step sc = step.atC(i);
                Step sd = step.atD(i);
                if (sa.valid()) Q.push(sa);
                if (sb.valid()) Q.push(sb);
                if (sc.valid()) Q.push(sc);
                if (sd.valid()) Q.push(sd);
            }
        }
        if (!found) cout << "Impossible" << endl;
    }

}
```

uva/12103.cpp

```
//12103
//Leonardo's Notebook
//Misc;Permutation Cycle
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

int main() {
    int t; cin >> t;
    while(t--) {
        string s; cin >> s;
```

```
13
14          int notzero = 0, visited = 0;
15          for(int i=0; i<s.size(); i++) {
16              if (visited & 1<<i) continue;
17
18              int cycle = 0;
19              for(int j=i; ~visited & 1<<j; j=s[j]-'A') {
20                  visited |= 1<<j;
21                  cycle++;
22              }
23              if (cycle % 2 == 0)
24                  notzero ^= 1<<cycle;
25          }
26
27          cout << (notzero ? "No" : "Yes") << endl;
28      }
29  }
```

uva/12135.cpp

```
1   //12135
2   //Switch Bulbs
3   //Graphs;Shortest Path;BFS
4   #include <iostream>
5   #include <queue>
6   #include <cstring>
7   #include <string>
8   #define MAX 33000
9   using namespace std;
10
11  vector<int> G[MAX];
12  int V[MAX];
13
14  int n, m;
15  struct Step {
16      int x, w;
17      Step() {}
18      Step(int x, int w) : x(x), w(w) {}
19  };
20
21  int main() {
22      int t; cin >> t; int tt=0;
23      while(cin >> n >> m, t--) {
24          memset(G, 0, sizeof(G));
25          memset(V, -1, sizeof(V));
26
27          n = 1<<n;
28
29          for(int i=0; i<m; i++) {
30              int a, b, mask=0;
31              cin >> a;
32              while(a--) {
33                  cin >> b;
34                  mask = mask | (1<<b);
35              }
36              for(int i=0; i<n; i++)
37                  G[i].push_back(i^mask);
38          }
39
40          queue<Step> Q;
41          Q.push(Step(0, 0));
42          while(!Q.empty()) {
43              Step step = Q.front(); Q.pop();
44              if (V[step.x] >= 0) continue;
45
46              V[step.x] = step.w;
47              for(int i=0; i<G[step.x].size(); i++)
48                  Q.push(Step(G[step.x][i], step.w+1));
49          }
50
51          cout << "Case " << ++tt << ":" << endl;
52          int q; string s;
53          cin >> q;
54          while(q--) {
55              int b = 0;
56              cin >> s;
57              for(int i=0; i<s.size(); i++)
58                  b = b*2 + (s[i]-'0');
59
60              cout << V[b] << endl;
61          }
```

```
        cout << endl;
    }

}
```

uva/12137.cpp

```
//12137
//Puzzles of Triangles
//Math;Prime Factorization
#include <string.h>
#include <stdio.h>
#define PP 20000
#define ull unsigned long long

int W[PP], wn=0;
bool P[PP];

inline ull div(const ull& a, const ull& b, ull &r) {
    r = a/b;
    return a-r*b;
}

inline ull pow(const ull& a, const int b) {
    if (b==0) return 1;
    ull tmp = b&1 ? a : 1;
    ull r = pow(a, b>>1);
    return tmp*r*r;
}

int main() {
    for(long long i=2; i*i<PP; i++) {
        if (P[i]) continue;
        W[wn++] = i;
        for(long long j=i*i; j<PP; j+=i) {
            P[j] = true;
        }
    }

    ull n;
    int t=0;
    while(scanf("%llu", &n), n) {
        ull ncopy = n;
        ull step = 1;
        for(int i=0; ncopy>1 && i<wn; i++) {
            int power=0;
            ull divr;
            while(div(ncopy, W[i], divr)==0) {
                ncopy = divr;
                power++;
            }
            step *= pow(W[i], (power+1)/2);
        }
        step *= ncopy;

        ull result;
        if (div(n, step, result)==0) result--;
        result *= 8;

        if(result)
            printf("Case %d: %llu\n",++t, result);
        else
            printf("Case %d: Impossible\n", ++t);
    }
}
```

uva/12144.cpp

```
//12144
//Almost Shortest Path
//Graphs;Shortest Path;Dijkstra
#include <iostream>
#include <cstring>
#include <climits>
#include <vector>
#include <algorithm>
#include <queue>
#define MAX 501

```

```cpp
using namespace std;

struct Edge {
    int u, v, c;
    Edge(int u, int v, int c) : u(u), v(v), c(c) {}
    inline bool operator < (const Edge& that) const { return c > that.c; }
};

int G[MAX][MAX];
int V[MAX];
vector<int> D[MAX];
int n, m, s, t;

void remove(int t) {
    if (D[t].size() == 0 || t == D[t][0]) return;
    for(int i=0; i<D[t].size(); i++) {
        G[D[t][i]][t] = 0;
        remove(D[t][i]);
    }
}

int shortest() {
    memset(V, 0x3f, sizeof(V));
    memset(D, 0, sizeof(D));
    priority_queue<Edge> Q;
    Q.push(Edge(s, s, 0));

    while(!Q.empty()) {
        Edge item = Q.top(); Q.pop();
        if (item.c > V[item.v]) continue;
        V[item.v] = item.c;
        D[item.v].push_back(item.u);

        for(int j=0; j<n; j++) {
            if (G[item.v][j]) {
                Edge e = Edge(item.v, j, item.c+G[item.v][j]);
                if (e.c <= V[e.v])
                    Q.push(e);
            }
        }
    }
    remove(t);
    if (V[t] < 0x3f3f3f3f)
        return V[t];
    else
        return -1;

}

int main() {
    while(cin >> n >> m, n|m) {
        cin >> s >> t;
        memset(G, 0, sizeof(G));

        for(int i=0; i<m; i++) {
            int a, b, c;
            cin >> a >> b >> c;
            G[a][b] = c;
        }

        shortest();
        cout << shortest() << endl;
    }
    return 0;
}
```

uva/12147.cpp

```cpp
//12147
//DNA Sequences
//Dynamic Programming;Longest Common Subsequence
#include <iostream>
#include <string>
#include <cstring>
#include <cmath>
#define MAX 1005
using namespace std;

int T[MAX][MAX];
int S[MAX][MAX];
string P, Q;
```

```
14
15  int main() {
16      int k;
17      while(cin >> k, k) {
18          cin >> P >> Q;
19          int p = P.size(), q = Q.size();
20
21          for(int i=0; i<=p; i++) T[i][0] = S[i][0] = 0;
22          for(int i=0; i<=q; i++) T[0][i] = S[0][i] = 0;
23
24          for(int i=1; i<=p; i++) {
25              for(int j=1; j<=q; j++) {
26                  if (P[i-1] == Q[j-1])
27                      S[i][j] = S[i-1][j-1] + 1;
28                  else
29                      S[i][j] = 0;
30              }
31          }
32
33          for(int i=1; i<=p; i++) {
34              for(int j=1; j<=q; j++) {
35                  T[i][j] = max(T[i-1][j], T[i][j-1]);
36
37                  for(int s=k; s<=S[i][j]; s++)
38                      T[i][j] = max(T[i][j], T[i-s][j-s]+s);
39              }
40          }
41          cout << T[p][q] << endl;
42      }
43
44      return 0;
45  }
```

uva/12148.cpp

```
1   //12148
2   //Electricity
3   //Misc;Ad hoc
4   #include <iostream>
5   using namespace std;
6
7   int M[] = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
8
9   bool oneday(int ad, int am, int ay, int bd, int bm, int by) {
10      if (--bd == 0) {
11          if (--bm == 0) {
12              --by;
13              bm=12;
14          }
15
16          bd = M[bm-1];
17
18          bool isleap = (by%4==0 && (by%100!=0 || by%400==0));
19          if (bm==2 && isleap) bd=29;
20      }
21      return ad==bd && am==bm && ay==by;
22
23  }
24
25  int main() {
26      int n, ad=0, am=0, ay=0, ac=0;
27      while(cin >> n, n) {
28          int sum = 0, count=0;
29          while(n--) {
30              int bd, bm, by, bc;
31              cin >> bd >> bm >> by >> bc;
32              if (oneday(ad, am, ay, bd, bm, by)) {
33                  sum += bc-ac; count++;
34              }
35              ad = bd; am = bm; ay = by; ac = bc;
36          }
37          cout << count << " " << sum << endl;
38      }
39
40      return 0;
41  }
```

uva/12155.cpp

```cpp
//12155
//ASCII Diamondi
//Misc;Ad hoc
#include <iostream>
using namespace std;

inline int abs(int n) { return n>0?n:-n; }

inline char charAt(int n, int x, int y) {
    x%=n*2-1; y%=n*2-1;
    int dist = abs(n-x-1)+abs(n-y-1);
    if (dist < n)
        return (char)(dist%26+'a');
    else
        return '.';
}

int main() {
    int n, ax, ay, bx, by, t=0;
    while(cin >> n, n) {
        cin >> ax >> ay >> bx >> by;
        cout << "Case " << ++t << ":" << endl;
        for(int i=ax; i<=bx; i++) {
            for(int j=ay; j<=by; j++) {
                cout << charAt(n, i, j);
            }
            cout << endl;
        }
    }

}
```

uva/12159.cpp

```cpp
//12159
//Gun Fight
//Graphs;Bipartite Matching
#include <iostream>
#include <iomanip>
#include <cstring>
#include <vector>
#include <cmath>
#include <climits>
#include <vector>
#include <cassert>
#define MAX 306
using namespace std;

int X[MAX], Y[MAX], P[MAX], G[MAX][MAX], n, r, a, b;
bool V[MAX];

bool team(int c) {
    return (X[b] - X[a])*(Y[c] - Y[a]) - (Y[b] - Y[a])*(X[c] - X[a]) > 0;
}

int sqrdist(int a, int b) {
    return (X[a]-X[b])*(X[a]-X[b])+(Y[a]-Y[b])*(Y[a]-Y[b]);
}

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=0; i<=n; i++) {
        if (!V[i] && G[s][i] > 0) {
            if (int sent = send(i, t, min(minn, G[s][i]))) {
                G[s][i] -= sent;
                G[i][s] += sent;
                return sent;
            }
        }
    }
    return 0;
}

int main() {
    int t=0;
    while(cin >> n, n) {
        memset(G, 0, sizeof(G));

        for(int i=1;i<=n;i++)
```

```
            cin >> X[i] >> Y[i] >> P[i];
        cin >> a >> b >> r;

        vector<int> A, B;
        for(int i=1;i<=n;i++) {
            if (P[i] == 0) continue;
            if (team(i))
                B.push_back(i);
            else
                A.push_back(i);
        }
        if (A.size() > B.size()) A.swap(B);

        for(int i=0; i<A.size(); i++) {
            int u=A[i];
            G[0][u] = 1;
            for(int j=0; j<B.size(); j++) {
                int v = B[j];
                G[v][n+1] = 1;
                if (sqrdist(u, v) <= r*r && P[u] > P[v])
                    G[u][v] = 1;
            }
        }
        n++;

        memset(V, 0, sizeof(V));
        int total = 0;
        while(int sent = send(0, n, INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }
        cout << "Case " << ++t << ": " << total << endl;
    }
}
```

uva/12160.cpp

```
//12160
//Unlock the Lock
//Graphs;Shortest Path;BFS
#include <iostream>
#include <queue>
#include <cstring>
#include <string>
using namespace std;

bool V[10000];
int R[10];

struct Step {
    int x, w;
    Step() {}
    Step(int x, int w) : x(x), w(w) {}

    Step sum(int n) {
        return Step((x+n)%10000, w+1);
    }
};

int main() {
    int a, b, n, t=0;
    while(cin >> a >> b >> n, a|b|n) {
        for(int i=0;i<n;i++)
            cin >> R[i];

        cout << "Case " << ++t << ": ";
        memset(V, 0, sizeof(V));
        queue<Step> Q;
        Q.push(Step(a, 0));
        bool found = false;
        while(!Q.empty()) {
            Step step = Q.front(); Q.pop();
            if (V[step.x]) continue;
            V[step.x] = true;
            if (step.x == b) {
                cout << step.w << endl;
                found = true;
                break;
            }
            for(int i=0;i<n;i++)
                Q.push(step.sum(R[i]));
```

```
        }
        if (!found) cout << "Permanently Locked" << endl;
    }

}
```

uva/12168.cpp

```
//12168
//Cat vs. Dog
//Graphs;Bipartite Matching;Konig Theorem
#include <iostream>
#include <cstring>
#include <climits>
#include <string>
#define MAX 505
using namespace std;

string V1[MAX], V2[MAX];
int G[MAX][MAX], n;
bool V[MAX];

int send(int s, int t, int minn) {
    V[s] = true;

    if (s==t) return minn;
    for(int i=0; i<=n; i++) {
        if (!V[i] && G[s][i] > 0) {
            if (int sent = send(i, t, min(minn, G[s][i]))) {
                G[s][i] -= sent;
                G[i][s] += sent;
                return sent;
            }
        }
    }
    return 0;
}

int main() {
    int t; cin >> t;
    int c, d, v;

    while(cin >> c >> d >> v, t--) {
        memset(G, 0, sizeof(G));
        memset(V, 0, sizeof(V));

        string s1, s2;
        for(int i=1; i<=v; i++) {
            cin >> s1 >> s2;
            V1[i] = s1; V2[i] = s2;

            bool dog = s1[0] == 'D';

            if (dog)
                G[0][i] = 1;
            else
                G[i][v+1] = 1;

            for(int j=1; j<i; j++) {
                if (s1 == V2[j] || s2 == V1[j])
                    if (dog)
                        G[i][j] = 1;
                    else
                        G[j][i] = 1;
            }
        }
        n = v+1;

        int total = 0;
        while(int sent = send(0, n, INT_MAX)) {
            total += sent;
            memset(V, 0, sizeof(V));
        }
        cout << v-total << endl;
    }

    return 0;
}
```

uva/12172.cpp

```cpp
//12172
//Matchsticks
//Misc;Greed
#include <iostream>
#include <cstring>
#include <cmath>
#define MAX 101
using namespace std;

void printMax(int n) {
    if (n&1) { cout << "7"; n-=3; }
    for(;n;n-=2) cout << "1";
}

void printMin(int n) {
    switch(n) {
        case 2: cout << "1"; return;
        case 3: cout << "7"; return;
        case 4: cout << "4"; return;
        case 5: cout << "2"; return;
        case 6: cout << "6"; return;
    }

    switch(n%7) {
        case 1: cout << "10"; n-=8; break;
        case 2: cout << "1"; n-=2; break;
        case 3:
            if (n==10) {
                cout << "22"; n-= 10;
            } else{
                cout << "200"; n-=17;
            }
            break;
        case 4: cout << "20"; n-= 11; break;
        case 5: cout << "2"; n-= 5; break;
        case 6: cout << "6"; n-= 6; break;
    }
    for(;n;n-=7) cout << "8";
}

int main() {


    int n;
    int t; cin >> t; t=0;

    while(cin >> n) {
        printMin(n);
        cout << " ";
        printMax(n);
        cout << endl;
    }

    return 0;
}
```

uva/12179.cpp

```cpp
//12179
//Randomly-priced Tickets
//Graphs;Shortest Path;Floyd-Warshall
#include <iostream>
#include <cstring>
#include <cmath>
#include <iomanip>
#define MAX 101
using namespace std;

int G[MAX][MAX], n, r, c;
double P[101][10001];

int main() {
    int t; cin >> t; t=0;
    cout << fixed << setprecision(6);

    while(cin >> n >> r) {
        memset(G, 0x3F, sizeof(G));
```

```
        memset(P, 0, sizeof(P));

        char cc;
        for(int i=0; i<n; i++) {
            for(int j=0; j<n; j++) {
                cin >> cc;
                if (cc=='Y') G[i][j] = 1;
            }
        }

        for(int k=0; k<n; k++)
            for(int i=0; i<n; i++)
                for(int j=0; j<n; j++)
                    G[i][j] = min(G[i][j], G[i][k] + G[k][j]);

        P[0][0] = 1;
        double pp = 1.0/r;
        for(int i=1; i<=100; i++)
            for(int k=1; k<=r; k++)
                for(int j=k; j<=100*r; j++)
                    P[i][j] += P[i-1][j-k] * pp;

        cout << "Case " << ++t << endl;
        cin >> c;
        while(c--) {
            int a, b, m;
            cin >> a >> b >> m;
            a--; b--;

            int d=G[a][b];

            double total = 0;
            for(int i=0; i<=m; i++)
                total += P[d][i];
            cout << total << endl;
        }
        cout << endl;
    }
    return 0;
}
```

uva/12184.cpp

```
//12184
//Transcribed Books
//Math;GCD
#include <iostream>
using namespace std;

long gcd(long a, long b) {
    while(b) {
        long c = a%b;
        a = b;
        b = c;
    }
    return a;
}

int main() {
    int t; cin >> t;
    int n;
    while(cin >> n) {
        long result = 0;
        long maxSerial = 0;
        for(int i=0; i<n; i++) {
            long s=0, tmp;
            for(int j=0; j<9; j++) {
                cin >> tmp; s+=tmp;
            }
            cin >> tmp;
            s -= tmp;
            maxSerial = max(maxSerial, tmp);
            result = gcd(result, s);
        }
        if (result>1 && maxSerial < result)
            cout << result << endl;
        else
            cout << "impossible" << endl;
    }
}
```

uva/12186.cpp

```cpp
//12186
//Another Crisis
//Graphs;DFS
#include <iostream>
#include <vector>
#include <algorithm>
#include <cstring>
#include <cmath>
#define MAX 100002
using namespace std;

vector<int> G[MAX];
int n, t;

int dfs(int v) {
    if (G[v].empty()) return 1;
    vector<int> mins;
    for(int i=0; i<G[v].size(); i++)
        mins.push_back(dfs(G[v][i]));
    sort(mins.begin(), mins.end());

    int get = (int)ceil(G[v].size()*t/100.0);
    int sum = 0;
    for(int i=0; i<get; i++) sum+=mins[i];
    return sum;
}

int main() {
    int boss;
    while(cin >> n >> t, n|t) {
        memset(G, 0, sizeof(G));
        for(int i=1; i<=n; i++) {
            cin >> boss; G[boss].push_back(i);
        }
        cout << dfs(0) << endl;
    }
    return 0;
}
```

uva/12189.cpp

```cpp
//12189
//Dinner Hall
//Misc;Sort
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

struct Event {
    int s; char t;
    Event() {}
    Event(int s, char t) : s(s), t(t) {}
    int entry() { return t=='E'?1:0; }
    int exit() { return t=='X'?1:0; }
    int unknown() { return t=='?'?1:0; }
};

bool compare(const Event& a, const Event& b) {
    return a.s < b.s;
}

vector<Event> V;

int main() {
    int n;
    while(cin >> n, n) {
        int entries=0, exits=0, unknowns=0;
        int a, b, c; char t;
        V.clear();
        for(int i=0; i<n; i++) {
            cin >> a >> t >> b >> t >> c >> t;
            Event e = Event(a*60*60+b*60+c, t);
            entries += e.entry();
            exits += e.exit();
            unknowns += e.unknown();
            V.push_back(e);
```

```
        }
        sort(V.begin(), V.end(), compare);

        int maxEntries = (unknowns-(entries-exits))/2;
        int maxx = 0, current=0;
        for(int i=0; i<V.size(); i++) {
            if (V[i].entry()) current++;
            if (V[i].exit()) current--;
            if (V[i].unknown()) {
                if (maxEntries) { current++; maxEntries--; }
                else { current--; }
            }
            maxx = max(maxx, current);
        }
        cout << maxx << endl;
    }
}
```

uva/12190.cpp

```
//12190
//Electric Bill
//Misc;Binary Search
#include <iostream>
using namespace std;

int C(int price) {
    int cons = 0;
    cons += min(max(0, price/2), 100); price -= 2*100;
    cons += min(max(0, price/3), 9900); price -= 3*9900;
    cons += min(max(0, price/5), 990000); price -= 5*990000;
    cons += max(0, price/7);
    return cons;
}

int V(int cons) {
    int price = 0;
    price += min(max(0, cons*2), 2*100); cons -= 100;
    price += min(max(0, cons*3), 3*9900); cons -= 9900;
    price += min(max(0, cons*5), 5*990000); cons -= 990000;
    price += max(0, cons*7);
    return price;
}

int main() {
    int a, b;
    while(cin >> a >> b,a|b) {
        int total = C(a);
        int begin = 0, end = total;
        int answer = 0;
        while(begin < end) {
            int mine = (begin+end)/2;
            int diff = V(total-mine)-V(mine);
            if (diff > b)
                begin = mine;
            else if (diff < b)
                end = mine;
            else { answer = mine; break; }
        }

        cout << V(answer) << endl;
    }

    return 0;
}
```

uva/12192.cpp

```
//12192
//Grapevine
//Misc;Binary Search
#include <iostream>
#include <cstring>
#include <vector>
#include <algorithm>
using namespace std;

int T[1001][501];
int S[1001];
```

```
12
13  int main() {
14      int n, m, q;
15      while(cin >> n >> m, n|m) {
16          memset(S, 0, (m+n)*sizeof(int));
17
18          for(int i=0; i<n; i++)
19              for(int j=0; j<m; j++)
20                  cin >> T[i-j+m][S[i-j+m]++];
21
22          cin >> q;
23          while(q--) {
24              int L, U;
25              cin >> L >> U;
26              int maxx = 0;
27              for(int i=0;i<m+n; i++) {
28                  int a = lower_bound(T[i], T[i]+S[i], L) - T[i];
29                  int b = upper_bound(T[i], T[i]+S[i], U) - T[i];
30                  maxx = max(maxx, b-a);
31              }
32              cout << maxx << endl;
33          }
34
35          cout << "-" << endl;
36      }
37  }
```

uva/12194.cpp

```
1   //12194
2   //Isosceles Triangles
3   //Math;Geometry
4   #include <cstdio>
5   #include <algorithm>
6   #include <cstring>
7   #define MAX 1010
8   using namespace std;
9
10  int X[MAX], Y[MAX];
11  long T[MAX][MAX];
12  int C[MAX];
13
14  inline long sqr(long v) { return v*v; }
15
16  int main(){
17      int n;
18      while(scanf("%d", &n), n) {
19          memset(C, 0, sizeof(C));
20
21          for(int i=0; i<n; i++)
22              scanf("%d %d", &X[i], &Y[i]);
23
24          int sum = 0;
25          for(int i=0; i<n; i++) {
26              for(int j=0; j<n; j++)
27                  T[i][C[i]++] = sqr(X[i]-X[j])+sqr(Y[i]-Y[j]);
28              sort(T[i], T[i]+C[i]);
29              long last=-1L;
30              int cnt=0;
31              for(int j=0; j<C[i]; j++) {
32                  if (T[i][j] != last) {
33                      sum += cnt*(cnt-1)/2;
34                      cnt = 0;
35                  }
36                  last = T[i][j];
37                  cnt++;
38              }
39              sum += cnt*(cnt-1)/2;
40          }
41
42          printf("%d\n", sum);
43      }
44
45  }
```

uva/12195.cpp

```
1   //12195
2   //Jingle Composing
```

```
3  //Misc;Ad hoc
4  #include <iostream>
5  #include <string>
6  using namespace std;
7
8  int duration(char c) {
9      switch(c) {
10         case 'W': return 64;
11         case 'H': return 32;
12         case 'Q': return 16;
13         case 'E': return 8;
14         case 'S': return 4;
15         case 'T': return 2;
16         case 'X': return 1;
17     }
18 }
19
20 int main() {
21     string s;
22     while(cin >> s, s!="*") {
23         int d=0, r=0;
24         for(int i=1; i<s.size(); i++) {
25             if (s[i] == '/') {
26                 if (d==64) r++;
27                 d = 0;
28                 continue;
29             }
30             d+=duration(s[i]);
31         }
32         cout << r << endl;
33     }
34
35     return 0;
36 }
```

uva/12196.cpp

```
1  //12196
2  //Klingon Levels
3  //Misc;Ad hoc
4  #include <iostream>
5  #include <climits>
6  #include <cstring>
7  using namespace std;
8
9  int T[10001][1001];
10 int N[10001];
11
12 inline long abs(long n) { return n>0?n:-n;}
13
14 int main() {
15     int n, tmp;
16     while(cin >> n, n) {
17         memset(T, 0, n*sizeof(T[0]));
18         for(int i=0; i<n; i++) {
19             cin >> N[i];
20             for(int j=0; j<N[i]; j++) {
21                 cin >> tmp;
22                 T[i][tmp]++;
23             }
24             for(int j=1;j<=1000;j++)
25                 T[i][j] += T[i][j-1];
26         }
27
28         long minn = INT_MAX;
29         for(int t=0;t<=1000;t++) {
30             long sum=0;
31             for(int i=0; i<n; i++) {
32                 sum += abs(N[i] - 2*T[i][t]);
33             }
34             minn = min(minn, sum);
35         }
36         cout << minn << endl;
37     }
38 }
```

uva/12300.cpp

```
1  //12300
```

```cpp
//Smallest Regular Polygon
//Math;Geometry
#include <iostream>
#include <cmath>
#include <iomanip>
#define PI 3.14159265358979323846
using namespace std;

double cot(double angle) {
    return cos(angle)/sin(angle);
}

int main(){
    int x1, y1, x2, y2, n;
    while(cin >> x1 >> y1 >> x2 >> y2 >> n, x1 | y1 | x2 | y2 | n) {
        double d = sqrt(pow(x2-x1, 2.0)+pow(y2-y1, 2.0));
        int k = n/2;
        double s = sin(PI/n)/sin(PI*k/n)*d;
        double A = 0.25*n*s*s*cot(PI/n);
        setprecision(6);
        cout << fixed << A << endl;
    }

}
```

uva/12361.cpp

```cpp
//12361
//File Retrieval
//Misc;String Matching;Suffix Array;Longest Common Prefix
#include <iostream>
#include <iomanip>
#include <cstring>
#include <string>
#include <sstream>
#include <cmath>
#include <set>
#include <stack>
#define MAX 600200
#define ull unsigned long long
using namespace std;

struct Item {
    ull v; int p;
    Item(ull v, int p) : v(v), p(p) { }
};

int RA[MAX], tempRA[MAX];
int SA[MAX], tempSA[MAX];
int C[MAX];
int Phi[MAX], PLCP[MAX], LCP[MAX];
int IDX[MAX], SIZ[MAX];
set<ull> R;

void suffix_sort(int n, int k) {
    memset(C, 0, sizeof C);

    for (int i = 0; i < n; i++)
        C[i + k < n ? RA[i + k] : 0]++;

    int sum = 0;
    for (int i = 0; i < max(256, n); i++) {
        int t = C[i];
        C[i] = sum;
        sum += t;
    }

    for (int i = 0; i < n; i++)
        tempSA[C[SA[i] + k < n ? RA[SA[i] + k] : 0]++] = SA[i];

    memcpy(SA, tempSA, n*sizeof(int));
}

void suffix_array(string &s) {
    int n = s.size();

    for (int i = 0; i < n; i++)
        RA[i] = s[i] - 1;

    for (int i = 0; i < n; i++)
        SA[i] = i;
```

```
55
56
57      for (int k = 1; k < n; k *= 2) {
58          suffix_sort(n, k);
59          suffix_sort(n, 0);
60
61          int r = tempRA[SA[0]] = 0;
62          for (int i = 1; i < n; i++) {
63              int s1 = SA[i], s2 = SA[i-1];
64              bool equal = true;
65              equal &= RA[s1] == RA[s2];
66              equal &= s1+k < n && s2+k < n && RA[s1+k] == RA[s2+k];
67
68              tempRA[SA[i]] = equal ? r : ++r;
69          }
70
71          memcpy(RA, tempRA, n*sizeof(int));
72      }
73  }
74
75  void lcp(string &s) {
76      int n = s.size();
77
78      Phi[SA[0]] = -1;
79      for (int i = 1; i < n; i++)
80          Phi[SA[i]] = SA[i-1];
81
82      int L = 0;
83      for (int i = 0; i < n; i++) {
84          if (Phi[i] == -1) {
85              PLCP[i] = 0;
86              continue;
87          }
88          while (s[i + L] != '\1' && s[i + L] == s[Phi[i] + L])
89              L++;
90
91          PLCP[i] = L;
92          L = max(L-1, 0);
93      }
94
95      for (int i = 1; i < n; i++)
96          LCP[i] = PLCP[SA[i]];
97  }
98
99  int main() {
100     int n;
101     while(cin >> n, n) {
102         R.clear();
103
104         stringstream ss; int kk = 0;
105         for(int i=0; i<n; i++) {
106             string temp;
107             cin >> temp;
108             ss << temp << '\1';
109
110             for(int j=0; j<=temp.size(); j++) {
111                 SIZ[kk] = temp.size()-j;
112                 IDX[kk] = i;
113                 kk++;
114             }
115         }
116
117         string s = ss.str();
118
119         suffix_array(s);
120         lcp(s);
121
122         stack<Item> ST;
123
124         for(int i=n; i<s.size(); i++) {
125             if (LCP[i] < SIZ[SA[i]] && (i+1==s.size() || LCP[i+1] < SIZ[SA[i]]))
126                 R.insert(1ull << IDX[SA[i]]);
127         }
128
129         for(int i=n; i<s.size(); i++) {
130             ull lastv = 0;
131             while(!ST.empty() && (ST.top().p > LCP[i] || LCP[i] == 0)) {
132                 Item item = ST.top(); ST.pop();
133
134                 R.insert(item.v);
135
136                 if (!ST.empty())
```

```
                    ST.top().v |= item.v;

                lastv = item.v;
            }
            if (LCP[i]) {
                if (ST.empty() || ST.top().p < LCP[i]) {
                    ST.push(Item(1ull << IDX[SA[i]] | 1ull << IDX[SA[i-1]] | lastv, LCP[i]));
                } else if (ST.top().p == LCP[i]) {
                    ST.top().v |= 1ull << IDX[SA[i]];
                }
            }
        }


        while(!ST.empty()) {
            Item item = ST.top(); ST.pop();

            R.insert(item.v);
            if (!ST.empty())
                ST.top().v |= item.v;
        }


        cout << R.size() << endl;
    }
}
```

uva/12363.cpp

```
//12363
//Hedge Mazes
//Graphs;Finding Bridges
#include <iostream>
#include <cstring>
#include <string>
#include <sstream>
#include <vector>
#include <algorithm>
#define MAX 10001
using namespace std;

int V[MAX], L[MAX], P[MAX], n, gpe;
vector<int> G[MAX];

inline int findset(int v) {
    if (P[v] != -1 && P[v] != v)
        return P[v] = findset(P[v]);
    return v;
}

inline int unionset(int x, int y) {
    int a = findset(x), b = findset(y);
    if (a<b) swap(a,b);
    P[b] = a;
}

void dfs(int u, int v) {
    V[v] = L[v] = ++gpe;

    for(int i = 0; i < G[v].size(); i++) {
        int w = G[v][i];
        if(!V[w]){
            dfs(v, w);
            L[v] = min(L[v], L[w]);

            if (L[w] > V[v])
                unionset(v, w);
        } else if(w != u) {
            L[v] = min(L[v], V[w]);
        }
    }
}

int main() {
    int m, q;
    while(cin >> n >> m >> q, n|m|q) {
        memset(G, 0, sizeof(vector<int>)*(n+1));
        memset(V, 0, sizeof(int)*(n+1));
        memset(L, 0, sizeof(int)*(n+1));
        memset(P, -1, sizeof(int)*(n+1));
        gpe = 0;
```

```
53
54          for(int i=0; i<m; i++) {
55              int a, b;
56              cin >> a >> b;
57              G[a].push_back(b);
58              G[b].push_back(a);
59          }
60
61          for(int i=0; i<n; i++)
62              if (!V[i])
63                  dfs(i, i);
64
65          for(int i=0; i<q; i++) {
66              int a, b;
67              cin >> a >> b;
68              cout << (findset(a)==findset(b) ? "Y" : "N") << endl;
69          }
70          cout << "-" << endl;
71      }
72  }
```

uva/12365.cpp

```
1   //12365
2   //Jupiter Atacks!
3   //Misc;Fenwick Tree
4   #include <iostream>
5   #include <cstring>
6   #define MAX 100100
7   #define ull long long
8   using namespace std;
9
10  struct Fenwick {
11      ull T[MAX];
12      int n;
13
14      Fenwick() {
15          clear(0);
16      }
17
18      void clear(int n) {
19          n++;
20          memset(T, 0, n*sizeof(ull));
21          this->n = n;
22      }
23
24      void adjust(int k, ull v, ull p) {
25          for (; k < n; k += (k&-k)) {
26              T[k] += v;
27              T[k] %= p;
28          }
29      }
30
31      void update(int k, ull v, ull p) {
32          ull current = rsq(k, k, p);
33          adjust(k, v-current, p);
34      }
35
36      ull rsq(int b, ull p) {
37          ull sum = 0;
38          for (; b; b -= (b&-b))
39              sum += T[b]%p;
40          return sum;
41      }
42
43      ull rsq(int a, int b, ull p) {
44          return (rsq(b, p) - rsq(a - 1, p) + p) % p;
45      }
46  };
47
48  ull pow(ull a, ull b, ull p) {
49      if (not b) return 1;
50      ull x = pow(a%p, b%p/2, p) % p;
51      x = (x*x)%p;
52      if (b%2) x = (x*a)%p;
53      return x;
54  }
55
56  ull euclid(ull a, ull b, ull& rx, ull& ry) {
57      if (!b) return rx=1, ry=0, a;
58
```

```
    ull q = a/b;
    ull x, y;
    ull g = euclid(b, a-q*b, x, y);
    return rx=y, ry=x-q*y, g;
}

ull invert(ull a, ull p) {
    ull inverse, temp;
    euclid(a, p, inverse, temp);
    return inverse;
}

Fenwick T;

int main() {
    int B, P, L, N;
    while(cin >> B >> P >> L >> N, B|P|L|N) {
        T.clear(L);
        for(int i=0; i<N; i++) {
            char cmd; ull a, b;
            cin >> cmd >> a >> b;
            if (cmd == 'E') {
                T.update(a, b*pow(B, L-a, P), P);
            } else {
                ull raw = T.rsq(a, b, P);
                ull base = pow(B, L-b, P);

                cout << ((raw*invert(base, P))%P+P)%P << endl;
            }
        }
        cout << "-" << endl;
    }

    return 0;
}
```

uva/12482.cpp

```
//12482
//Short Story Competition
//Misc;Ad hoc
#include <iostream>
#include <cmath>
#include <string>
using namespace std;

int main() {
    int n, li, c;
    while(cin >> n >> li >> c) {
        string s;
        int pag = 1;
        int car = 0;
        for(int i=0; i<n; i++) {
            if (car > 0) car++;

            cin >> s;
            if (car + s.size() > c) {
                pag++;
                car = s.size();
            } else {
                car += s.size();
            }
            //cout << s<< " " << pag << " " << car << endl;
        }
        cout << ceil(pag/(double)li) << endl;



    }
}
```

uva/12483.cpp

```
//12483
//Toboggan of Marbles
//Math;Geometry;Point to Line
#include <iostream>
#include <cmath>
#include <string>
```

```cpp
#include <cstring>
#include <iomanip>
#define ull unsigned long long;
using namespace std;

struct Point {
    double x, y;

    Point() {}
    Point(double x, double y) : x(x), y(y) {}

    double dist(Point A) {
        return sqrt(pow(A.x-x,2)+pow(A.y-y,2));
    }

    double toLine(Point A, Point B) {
        double scale = ((x - A.x) * (B.x - A.x) + (y-A.y)*(B.y-A.y)) /
                       ((B.x - A.x) * (B.x - A.x) + (B.y-A.y)*(B.y-A.y));

        return dist(Point(A.x + scale*(B.x-A.x), A.y + scale * (B.y-A.y)));
    }

    double toSegment(Point A, Point B) {
        if ((x - A.x) * (B.x - A.x) + (y-A.y)*(B.y-A.y) <= 1e-6)
            return dist(A);

        if ((x - B.x) * (A.x - B.x) + (y-B.y)*(A.y-B.y) <= 1e-6)
            return dist(B);

        return toLine(A, B);
    }
};

int main() {
    int n, L, H;
    while(cin >> n >> L >> H) {
        Point pa, pb;
        int ya, yb;

        double minn = 100000000.0;
        for(int i=0;i<n;i++) {
            cin >> ya >> pa.x >> pa.y;

            int lado = i&1?L:0;
            int outroLado = i&1?0:L;

//            cout << " " << pa.x << " " << pa.y << "^" << outroLado;
            minn = min(minn, pa.toSegment(Point(outroLado, 0), Point(outroLado, H)));

            //cout << " " << pa.toSegment(Point(outroLado, 0), Point(outroLado, H));
            if (i>0) {
                minn = min(minn, pb.toSegment(Point(lado, ya), pa));
    //            cout << " " << pb.toSegment(Point(lado, ya), pa);
            }
            //cout << endl;


            yb = ya;
            pb = pa;
        }
        cout << fixed << setprecision(2) << minn << endl;

    }

}
```

uva/12484.cpp

```cpp
//12484
//Cards
//Dynamic Programming;Minimax
#include <iostream>
#include <cstring>
#include <algorithm>
#define ull long long
using namespace std;

ull T[10006], Q[10006], M[10006];


int main() {
```

```
    int n;
    while(cin >> n) {
        for(int i=1; i<=n; i++) {
            cin >> M[i];
            M[i]+=M[i-1];
        }

        memset(T, 0, sizeof(ull)*n);

        for(int i=1; i<=n; i++) {
            for(int j=0; j<=n-i; j++)
                Q[j] = M[j+i]-M[j] - min(T[j], T[j+1]);

            swap(T, Q);
        }

        cout << T[0] << endl;
    }

}
```

uva/12485.cpp

```
//12485
//Perfect Choir
//Misc;Ad hoc
#include <iostream>
#include <cmath>
#include <cstring>
using namespace std;

int T[10005];

int main() {
    int n;
    while(cin >> n) {
        memset(T, 0, sizeof(T));
        int total = 0;
        for(int i=0; i<n; i++) {
            int a; cin >> a;
            total += a;
            T[i] = a;
        }

        if (total % n != 0) {
            cout << -1 << endl;
            continue;
        }

        int media = total / n;
        int maior = 0;
        for(int i=0; i<n; i++) {
            if (T[i] > media)
                maior += T[i] - media;
        }

        cout << maior +1 << endl;
    }
}
```

uva/12486.cpp

```
//12486
//Space Elevator
//Misc;Binary Search
#include <iostream>
#include <cmath>
#include <string>
#include <cstring>
#include <iomanip>
#include <vector>
#include <algorithm>
#include <cstdio>
#define ull unsigned long long int
using namespace std;

ull T[20][10];

bool has(ull n, ull k, ull p) {
```

```
18      while(n) {
19              if (n%p==k) return true;
20              n/=10;
21      }
22      return false;
23  }
24
25  bool has(ull n) {
26      return has(n, 13, 100) || has(n, 4, 10);
27  }
28
29  ull right(ull n) {
30      int log10 = 0;
31      ull right = 0;
32
33      if (!has(n))
34          right++;
35
36      while(n) {
37          ull hi = n/10;
38          ull lo = n%10;
39
40          if (!has(hi)) {
41              for(ull i=0; i<lo; i++) {
42                  if (i!=4 && (hi%10 != 1 || i!=3))
43                      right += T[log10][i];
44              }
45          }
46
47          log10++;
48          n/=10;
49      }
50
51      return right-1;
52  }
53
54  ull answer(ull n) {
55      ull begin=0, end=-1;
56
57      while(begin+1 < end) {
58          ull mid = begin + (end - begin)/2;
59          ull v = right(mid);
60
61          if (v>=n)
62              end = mid;
63          else
64              begin = mid;
65      }
66
67      if (right(begin) == n)
68          return begin;
69      else
70          return end;
71  }
72
73  int main() {
74      for(ull i=0; i<10; i++) {
75          if (i==4) continue;
76          T[0][i] = 1;
77      }
78
79      for(ull i=1; i<20; i++) {
80          for(ull j=0; j<10; j++) {
81              if (j==4) continue;
82              for(ull k=0; k<10; k++) {
83                  if (k==3 and j==1) continue;
84                  T[i][j] += T[i-1][k];
85              }
86          }
87      }
88
89
90      ull n;
91      while(cin >> n) {
92         cout << answer(n) << endl;
93      }
94  }
```

uva/12487.cpp

```
1   //12487
```

```cpp
//Midnight Cowboy
//Graphs;Markov Chain
#include <iostream>
#include <cmath>
#include <string>
#include <cstring>
#include <iomanip>
#define ull unsigned long long;
using namespace std;

int G[101][101], S[101];
double M[101], Q[101];;

int main() {
    int n, a, b, c;
    while(cin >> n >> a >> b >> c) {
        memset(M, 0, sizeof(M));
        memset(S, 0, sizeof(S));
        memset(G, 0, sizeof(G));

        for(int i=0;i<n-1;i++) {
            int a, b; cin >> a >> b;
            G[a][S[a]++] = b;
            G[b][S[b]++] = a;
        }

        M[a] = 1.0;
        for(int k=0;k<10000;k++) {
            memset(Q, 0, sizeof(Q));
            Q[b] = M[b];
            Q[c] = M[c];
            for(int i=1;i<=n;i++) {
//                cout << M[i] << " ";
                if (i==b || i==c) continue;

                for(int j=0;j<S[i];j++)
                    Q[G[i][j]] += M[i] * 1.0/S[i];
            }
    //        cout << endl;

            swap(Q, M);
        }

        cout << fixed << setprecision(6) << M[b] << endl;
    }
}
```

uva/12488.cpp

```cpp
//12488
//Start Grid
//Misc;Ad hoc
#include <iostream>
#include <cstring>
#include <cmath>
using namespace std;

int L[30], C[30];

int abs(int n) {
    if (n < 0) return -n;
    return n;
}

int main() {
    int n;
    while(cin >> n) {
        for(int i=0; i<n; i++)
            cin >> L[i];

        for(int i=0; i<n; i++)
            cin >> C[i];

        int total = 0;
        for(int i=0; i<n; i++) {
            int cara = C[i];
            for(int j=n-1; j>i; j--) {
                if (L[j]==cara) {
                    int t = L[j];
                    L[j] = L[j-1];
                    L[j-1] = t;
```

```
                    total++;
                }
            }
        }

        cout << total << endl;
    }
}
```

uva/12489.cpp

```
//12489
//Combating cancer
//Graphs;Tree Isomorphism
#include <iostream>
#include <cmath>
#include <string>
#include <cstring>
#include <iomanip>
#include <vector>
#include <algorithm>
#define MAX 10006
using namespace std;

vector<int> A[MAX], B[MAX];
vector<int> NA[MAX], NB[MAX];

bool comp(const vector<int>& a, const vector<int>& b) {
    if (a.size() != b.size()) return a.size() < b.size();
    for(int i=0;i<a.size(); i++) {
        if (a[i] != b[i]) return a[i] < b[i];
    }
    return false;
}

bool eq(const vector<int>& a, const vector<int>& b) {
    if (a.size() != b.size()) return false;
    for(int i=0;i<a.size(); i++) {
        if (a[i] != b[i]) return false;
    }
    return true;
}

int main() {
    int n;
    while(cin >> n) {
        memset(A, 0, sizeof(A));
        memset(B, 0, sizeof(B));
        memset(NA, 0, sizeof(NA));
        memset(NB, 0, sizeof(NB));
        for(int i=0;i<n-1; i++) {
            int a, b; cin >> a >> b;
            A[a].push_back(b);
            A[b].push_back(a);
        }
        for(int i=0;i<n-1; i++) {
            int a, b; cin >> a >> b;
            B[a].push_back(b);
            B[b].push_back(a);
        }


        for(int i=1;i<=n; i++) {
            for(int j=0; j<A[i].size(); j++)
                NA[i].push_back(A[A[i][j]].size());
            sort(NA[i].begin(), NA[i].end());

            for(int j=0; j<B[i].size(); j++)
                NB[i].push_back(B[B[i][j]].size());
            sort(NB[i].begin(), NB[i].end());
        }

        sort(NA+1, NA+n+1, comp);
        sort(NB+1, NB+n+1, comp);


        bool equals = true;
        //cout << NA[n].size() << " " << NA[n].size() << " " << n << endl;
        for(int i=1; i<=n; i++) {
            // cout << i << " => ";
            // for(int j=0; j<NA[i].size(); j++) {
```

```
                // cout << NA[i][j] << " ";
            // }
            // cout << " | ";
            // for(int j=0; j<NB[i].size(); j++) {
                // cout << NB[i][j] << " ";
            // }
            // cout << endl;

            equals &= eq(NA[i], NB[i]);
        }
        cout << (equals ? "S" : "N") << endl;
    }

}
```

uva/12490.cpp

```
//12490
//Integral
//Misc;Ad hoc
#include <iostream>
#include <string>
#include <algorithm>
#include <cmath>
#define MAX 1000006
#define ull long long
using namespace std;

struct Value {
    int x;
    ull v;

    inline bool operator <(const Value& a) const {
        return this->x < a.x;
    }
};

Value F[MAX];

int main() {
    int n, s, y;
    while(cin >> n >> s >> y) {
        for(int i=0; i<s; i++) {
            cin >> F[i].x >> F[i].v;
        }

        sort(F, F+s);

        double minn = 0, maxx = 0;

        for(int i=0; i<s-1; i++) {
            Value a = F[i], b = F[i+1];

            minn += min(a.v, b.v) + (b.x - a.x-1) * min(a.v, b.v) + abs(a.v - b.v)/2.0;
            maxx += min(a.v, b.v) + (b.x - a.x-1) * max(a.v, b.v) + abs(a.v - b.v)/2.0;
        }

        if (y < minn || y > maxx || ceil(minn) != minn) {
            cout << "N" << endl; continue;
        }

        cout << "S";

        ull current = round(maxx);
        for(int i=0; i<s-1; i++) {
            Value a = F[i], b = F[i+1];

            ull delta = (b.x - a.x-1) * (max(a.v, b.v) - min(a.v, b.v));

            if (current == y) {
                for(int x = a.x+1; x<b.x; x++)
                    cout << " " << max(a.v, b.v);
            } else if (current - delta > y) {
                current -= delta;

                for(int x = a.x+1; x<b.x; x++)
                    cout << " " << min(a.v, b.v);
            } else if (a.v < b.v) {
                for(int x=a.x+1; x<b.x; x++) {
                    ull value = max(a.v, b.v - (current - y));
                    cout << " " << value;
```

```
                    current -= b.v - value;
                }
            } else {
                ull plus = (current - y) / (b.x - a.x - 1);
                ull rem = (current - y) % (b.x - a.x - 1);
                for(int x=a.x+1; x<b.x; x++) {
                    ull value = a.v - plus - (b.x - x <= rem ? 1: 0);
                    cout << " " << value;
                    current -= a.v - value;
                }
            }
        }
        cout << endl;

    }
}
```

uva/12491.cpp

```
//12491
//Words
//Misc;STL map
#include <iostream>
#include <string>
#include <map>
#include <set>
#define mit multimap<string, string>::iterator
#define mmit pair<multimap<string, string>::iterator,multimap<string, string>::iterator>
using namespace std;

multimap<string, string> X[2];
set<string> S[2], E[2];
char T[1000];

bool backtrack(int x, int k, int n) {
    string suffix(T+k, n-k);
    if (S[x].find(suffix) != S[x].end()) return false;
    S[x].insert(suffix);

    for(int s=1; s<=suffix.size(); s++) {
        string word = suffix.substr(0, s);

        for(int i=0; i<word.size(); i++)
            T[k+i] = word[i];

        if (E[x].find(word) != E[x].end()) {
            if (k + word.size()==n) return true;
            if (backtrack(x, k+word.size(), n)) return true;
        }

    }

    mmit ret = X[x].equal_range(suffix);
    for(mit it=ret.first;it != ret.second;it++) {
        string word = it->second;
        for(int i=0; i<word.size(); i++)
            T[k+i] = word[i];

        if (k + word.size() == n) return true;
        if (backtrack(1-x, n, k+word.size())) return true;
    }
    return false;
}

int main() {
    int a, b;
    while(cin >> a >> b) {
        E[0].clear(); E[1].clear();
        X[0].clear(); X[1].clear();
        S[0].clear(); S[1].clear();
        for(int i=0; i<a; i++) {
            string s; cin >> s;
            for(int j=0;j<=s.size(); j++) {
                E[0].insert(s);
                X[0].insert(pair<string, string>(s.substr(0, j), s));
            }
        }
        for(int i=0; i<b; i++) {
            string s; cin >> s;
            for(int j=0;j<=s.size(); j++) {
                E[1].insert(s);
```

```
                X[1].insert(pair<string, string>(s.substr(0, j), s));
            }
        }
        cout << (backtrack(0,0,0) || backtrack(1,0,0) ? 'S' : 'N') << endl;
    }

}
```

uva/12492.cpp

```
//12492
//Rubik Cycle
//Misc;Ad hoc
#include <iostream>
#include <string>
#define MAX 200010
using namespace std;

int T[54];

void rotate(int a, int b, int c, int d, int e, int f, int g, int h) {
    int x = T[h], y = T[g];
    T[h] = T[f];
    T[g] = T[e];

    T[f] = T[d];
    T[e] = T[c];

    T[d] = T[b];
    T[c] = T[a];

    T[b] = x;
    T[a] = y;
}

void adjust(int a, int b, int c, int d, int e, int f, int g, int h, int i, int j, int k, int l) {
    int x = T[j], y = T[k], z = T[l];

    T[j] = T[g];
    T[k] = T[h];
    T[l] = T[i];

    T[g] = T[d];
    T[h] = T[e];
    T[i] = T[f];

    T[d] = T[a];
    T[e] = T[b];
    T[f] = T[c];

    T[a] = x;
    T[b] = y;
    T[c] = z;
}

void F() {
    rotate(0, 1, 2, 5, 8, 7, 6, 3);
    adjust(33, 34, 35, 45, 48, 51, 11, 10, 9, 44, 41, 38);
}

void B() {
    rotate(26, 25, 24, 21, 18, 19, 20, 23);
    adjust(29, 28, 27, 36, 39, 42, 15, 16, 17, 53, 50, 47);
}

void L() {
    rotate(36, 37, 38, 41, 44, 43, 42, 39);
    adjust(0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33);
}

void R() {
    rotate(45, 46, 47, 50, 53, 52, 51, 48);
    adjust(8, 5, 2, 35, 32, 29, 26, 23, 20, 17, 14, 11);
}

void U() {
    rotate(27, 28, 29, 32, 35, 34, 33, 30);
    adjust(2, 1, 0, 38, 37, 36, 24, 25, 26, 47, 46, 45);
}

void D() {
```

```
    rotate(9, 10, 11, 14, 17, 16, 15, 12);
    adjust(6, 7, 8, 51, 52, 53, 20, 19, 18, 42, 43, 44);
}


bool ok() {
    for(int i=0; i<54; i++) {
        if (T[i] != i) return false;
    }
    return true;
}

int main() {
    string s;
    while(cin >> s) {
        for(int i=0; i<54; i++)
            T[i] = i;

        int result = 0;
        do {
            for(int i=0;i<s.size();i++) {
                switch(s[i]) {
                    case 'F': F(); break;
                    case 'B': B(); break;
                    case 'R': R(); break;
                    case 'L': L(); break;
                    case 'U': U(); break;
                    case 'D': D(); break;
                    case 'f': F(); F(); F(); break;
                    case 'b': B(); B(); B(); break;
                    case 'r': R(); R(); R(); break;
                    case 'l': L(); L(); L(); break;
                    case 'u': U(); U(); U(); break;
                    case 'd': D(); D(); D(); break;
                }
            }
            result++;
        } while (!ok());
        cout << result << endl;
    }
}
```

uva/12493.cpp

```
//12493
//Stars
//Math;Prime Factorization;Euler's Totient
#include <iostream>
#include <cmath>
#define PP 100000
#define ull unsigned long long
using namespace std;

bool P[PP];

int main() {
    for(long long i=2; i*i<PP; i++) {
        if (P[i]) continue;
        for(long long j=i*i; j<PP; j+=i) {
            P[j] = true;
        }
    }

    ull n;
    while(cin >> n) {
        ull tot = 1;
        //cout << ">>" << n << endl;
        for(ull i=2; i*i<=n && n>1; i++) {
            if (P[i]) continue;
            ull q=0;

            while(n%i==0) {
                n/=i;
                q++;
            }

            //if (q>0)
                //cout << i << " " << q << endl;
            if (q>0)
                tot *= (i-1) * (ull)pow(i, q-1);
        }
```

```
        if (n>1)
            tot *= n-1;

        cout << tot/2 << endl;

    }
}
```

uva/12506.cpp

```
//12506
//Shortest Names
//Misc;String Matching;Trie
#include <iostream>
#include <cstring>
#define MAXS 1000010
using namespace std;

struct Trie {
    int G[MAXS][26];
    int S[MAXS];
    int stateCount;

    Trie() {
        clear();
    }

    void clear() {
        stateCount = 0;
        clear(stateCount++);
    }

    int clear(int state) {
        memset(G[state], -1, sizeof G[state]);
        S[state] = 0;
        return state;
    }

    void add(string &s) {
        int state = 0;
        for(int i=0; i<s.size(); i++) {
            S[state]++;

            int next = s[i] - 'a';

            if (G[state][next] < 0)
                G[state][next] = clear(stateCount++);

            state = G[state][next];
        }
    }
};

Trie T;

int dfs(int state) {
    if (T.S[state] == 1) return 0;

    int s = T.S[state];

    for (int e = 0; e < 26; ++e) {
        if (T.G[state][e] == -1) continue;

        s += dfs(T.G[state][e]);
    }

    return s;
}


int main() {
    int tt; cin >> tt;
    while(tt--) {
        T.clear();

        int n; cin >> n;
        for(int i=0; i<n; i++) {
            string s; cin >> s;
            T.add(s);
        }

```

```
        cout << dfs(0) << endl;
    }
}
```

timus/1017.cpp

```
//1017
//Staircases
//Dynamic Programming;Ad hoc
#include <iostream>
#define MAX 506
using namespace std;

long long T[MAX][MAX];

int main() {
    for(int i=1;i<MAX; i++) {
        for(int j=1; j<=i; j++) {
            T[i][j] = 1;
            for(int k=j; k<=i; k++)
                T[i][j] += T[i-k][k+1];
        }
    }

    int n;
    while(cin >> n)
        cout << T[n][1]-1 << endl;
}
```

timus/1018.cpp

```
//1018
//Binary Apple Tree
//Dynamic Programming;Ad hoc
#include <iostream>
#include <vector>
#include <cstring>
#define MAX 105
using namespace std;

struct Node {
    int x, v;
    Node(int x, int v) : x(x), v(v) {}
};
vector<Node> T[MAX];
int S[MAX], TT[MAX][MAX];
bool V[MAX][MAX];

void adjust(int root, int parent) {
    for(int i=0; i<T[root].size(); i++) {
        Node node = T[root][i];
        if (node.x != parent) {
            adjust(node.x, root);
            S[node.x] = node.v;
        } else {
            T[root].erase(T[root].begin()+i);
            i--;
        }
    }
}

int answer(int root, int branches) {
    if (V[root][branches])
        return TT[root][branches];

    if (branches == 0) return 0;
    if (branches == 1) return S[root];
    if (T[root].size() == 0) return S[root];
    if (T[root].size() == 1) return S[root] + answer(T[root][0].x, branches-1);

    Node left = T[root][0];
    Node right = T[root][1];

    int maxx = 0;
    for(int i=0; i<=branches-1; i++)
        maxx = max(maxx, S[root] + answer(left.x, i) + answer(right.x, branches-1-i));

    V[root][branches] = true;
    return TT[root][branches] = maxx;
```

```
}

int main() {
    int n, q;
    while(cin >> n >> q) {
        memset(T, 0, sizeof(vector<int>)*n);
        memset(V, 0, sizeof(vector<int>)*n);
        memset(S, 0, sizeof(S));

        for(int i=0; i<n-1; i++) {
            int a, b, c; cin >> a >> b >> c;
            T[b].push_back(Node(a,c));
            T[a].push_back(Node(b,c));
        }
        adjust(1, -1);
        cout << answer(1, q+1) << endl;
    }
}
```

timus/1020.cpp

```
//1020
//Rope
//Math;Geometry
#include <iostream>
#include <cmath>
#include <iomanip>
#define PI 3.14159265
using namespace std;

struct Point {
    double x, y;

    Point() {}

    double dist(Point A) {
        return sqrt(pow(A.x-x,2)+pow(A.y-y,2));
    }
};

int main() {
    int n;
    double r;
    Point a, b, c;
    while(cin >> n >> r) {
        double total = 0;

        cin >> a.x >> a.y;
        b = a;
        for(int i=1; i<n; i++) {
            cin >> c.x >> c.y;
            total += b.dist(c);
            swap(b, c);
        }
        total += a.dist(b);

        cout << fixed << setprecision(2) << total + 2*PI*r << endl;
    }
}
```

timus/1030.cpp

```
//1030
//Titanic
//Math;Geometry;Great-Circle Distance
#include <iostream>
#include <cmath>
#include <string>
#include <iomanip>
#define PI 3.14159265
using namespace std;

double distance(double r, double x1, double y1, double x2, double y2) {
    return r*acos(sin(x1)*sin(x2) + cos(x1)*cos(x2)*cos(fabs(y1-y2)));
}

double readCoord() {
    int a1, a2, a3; char c; string s;
    cin >> a1 >> c >> a2 >> c >> a3 >> c >> s;
```

```
    double ret = (a1 + a2/60.0 + a3/3600.0) / 180.0 * PI;
    if (s=="WL." || s=="SL")
        ret = -ret;
    return ret;
}

double round(double d)
{
  return floor(d + 0.5);
}

int main() {
    string s;

    getline(cin, s);
    getline(cin, s);
    getline(cin, s);

    double X1 = readCoord();
    cin >> s;
    double Y1 = readCoord();
    getline(cin, s);
    getline(cin, s);
    double X2 = readCoord();
    cin >> s;
    double Y2 = readCoord();

    double d = distance(6875.0/2, X1, Y1, X2, Y2);
    cout << "The distance to the iceberg: " << fixed << setprecision(2) << round(d*100)/100.0 << " miles." << endl;
    if (d < 99.995)
        cout << "DANGER!" << endl;

}
```

timus/1111.cpp

```
//1111
//Squares
//Math;Geometry;Square Distance
#include <iostream>
#include <cmath>
#include <vector>
#include <algorithm>
using namespace std;

struct Point {
    double x, y;

    Point() {}
    Point(double x, double y) : x(x), y(y) {}

    double dist(const Point A) const {
        return sqrt(pow(A.x-x,2)+pow(A.y-y,2));
    }

    double toLine(const Point A, const Point B) const {
        double scale = ((x - A.x) * (B.x - A.x) + (y-A.y)*(B.y-A.y)) /
                       ((B.x - A.x) * (B.x - A.x) + (B.y-A.y)*(B.y-A.y));

        return dist(Point(A.x + scale*(B.x-A.x), A.y + scale * (B.y-A.y)));
    }

    double toSegment(const Point A, const Point B) const {
        if ((x - A.x) * (B.x - A.x) + (y-A.y)*(B.y-A.y) <= 1e-6)
            return dist(A);

        if ((x - B.x) * (A.x - B.x) + (y-B.y)*(A.y-B.y) <= 1e-6)
            return dist(B);

        return toLine(A, B);
    }

    int signal(const Point& a, const Point& b) const  {
        double sig = (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
        if (abs(sig) < 1e-6) return 0;
        if (sig < 0) return -1;
        return 1;
    }

    Point rotateWith(const Point origin, double si, double co, double scale) const {
```

```
        double tx = this->x - origin.x;
        double ty = this->y - origin.y;
        double x = (tx * co + ty * si)/scale;
        double y = (tx * -si + ty * co)/scale;
        return Point(origin.x + x, origin.y + y);

    }
};

struct Square {
    int id;
    Point a, b, c, d;
    Square(int id, Point x, Point y) : id(id),
        a(x.x, x.y), b(x.rotateWith(y, 0.707106781, 0.707106781, 1.41421356)),
        c(y.x, y.y), d(x.rotateWith(y, -0.707106781, 0.707106781, 1.41421356)) {}

    bool inside(const Point p) const {
        int sig = a.signal(p, b);
        if (sig == 0) return false;
        if (sig != b.signal(p, c)) return false;
        if (sig != c.signal(p, d)) return false;
        if (sig != d.signal(p, a)) return false;
        return true;
    }

    double dist(const Point p) const {
        if (inside(p)) return 0.0;
        return min(min(p.toSegment(a,b), p.toSegment(b, c)), min(p.toSegment(c,d), p.toSegment(d,a)));
    }
};

struct DistToP {
    Point p;
    DistToP(Point p) : p(p) {}

    inline bool operator() (const  Square &a, const  Square &b)  {
        double da = a.dist(this->p), db=b.dist(this->p);
        if (abs(da-db) > 1e-6) return da<db;
        return a.id < b.id;
    }
};

vector<Square> V;

int main() {
    int n;
    cin >> n;
    for(int i=1; i<=n; i++) {
        Point x,y; cin >> x.x >> x.y >> y.x >> y.y;
        V.push_back(Square(i, x, y));
    }

    Point p;
    cin >> p.x >> p.y;

    sort(V.begin(), V.end(), DistToP(p));
    for(int i=0;i<n; i++) {
        if (i) cout << " ";
        cout << V[i].id;
    }
    cout << endl;
}
```

timus/1159.cpp

```
//1185
//Wall
//Math;Geometry;Enclosing Circle
#include <iostream>
#include <cmath>
#include <iomanip>
#include <algorithm>
#define PI 3.14159265
#define EP 1e-10
using namespace std;

struct Point {
    long double x, y;

    Point() {}
    Point(long double x, long double y) : x(x), y(y) {}
```

```
};

int P[107];

long double simpleCase(long double maxx, int n) {
    long double begin=maxx/2.0, end=50000001.0;
    while(abs(begin-end) > EP) {
        long double r = (begin+end)/2;
        long double angle = 0;
        long double sum = 0;
        Point A(r,0);
        for(int i=0; i<n; i++) {
            angle += 2*asin(P[i]/(2.0*r));
            Point B(r*cos(angle), r*sin(angle));
            sum += (A.x + B.x) * (B.y - A.y);
            A = B;
        }
        sum /= 2;

        if (abs(angle-2*PI) < 1e-4)
            return sum;

        if (angle < 2*PI)
            end = r;
        else
            begin = r;
    }
    return 0.0;
}
long double complexCase(long double maxx, int n) {
    long double begin=maxx/2.0, end=50000001.0;
    while(abs(begin-end) > EP) {
        long double r = (begin+end)/2;
        long double angle = 2*asin(P[0]/(2.0*r));
        Point A(r*cos(angle),r*sin(angle));
        long double sum = (r + A.x) * (0 - A.y);
        for(int i=1; i<n; i++) {
            angle -= 2*asin(P[i]/(2.0*r));
            Point B(r*cos(angle), r*sin(angle));
            sum += (A.x + B.x) * (A.y - B.y);
            A = B;
        }
        sum /= 2;

        if (abs(angle) < EP)
            return sum;

        if (angle < 0)
            end = r;
        else
            begin = r;
    }
    return 0.0;
}

bool comp(int a, int b) {
    return a>b;
}

int main() {
    int n;
    while(cin >> n) {
        int maxx = 0, summ = 0;
        for(int i=0; i<n; i++) {
            cin >> P[i];
            maxx = max(maxx, P[i]);
            summ += P[i];
        }

        sort(P, P+n, comp);
        cout << fixed << setprecision(2) << max(simpleCase(maxx, n), complexCase(maxx, n)) << endl;

    }
}
```

timus/1185.cpp

```
//1185
//Wall
//Math;Geometry;Convex Hull;Monotone Chain
#include <iostream>
```

```cpp
#include <cmath>
#include <iomanip>
#include <algorithm>
#define PI 3.14159265
using namespace std;

struct Point {
    int x, y;

    Point() {}
    Point(int x, int y) : x(x), y(y) {}

    bool left(Point& a, Point& b) {
        return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x) < 0;
    }

    bool operator <(const Point& p) const {
        if (this->x != p.x) return this->x < p.x;
        return this->y < p.y;
    }

    bool operator ==(const Point& p) const {
        return this->x == p.x and this->y == p.y;
    }

    double dist(Point A) {
        return sqrt(pow(A.x-x,2.0)+pow(A.y-y,2.0));
    }
};

int convexHull(Point* P, int n, Point* S) {
    sort(P, P+n);

    int m=0;
    for(int i=0; i<n; i++) {
        while(m >= 2 && S[m-1].left(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    for(int i=n-1, k=m; i >= 0; i--) {
        while(m >= k+2 && S[m-1].left(S[m-2], P[i])) m--;
        S[m++] = P[i];
    }
    m--;

    return m;
}

Point P[1010], S[1010];

int main() {
    int n, r;
    while(cin >> n >> r) {
        for(int i=0; i<n; i++)
            cin >> P[i].x >> P[i].y;

        int s = convexHull(P, n, S);

        double total = 0;
        for(int i=0; i<s; i++)
            total += S[i].dist(S[(i+1)%n]);

        cout << floor(total + 2*PI*r + 0.5) << endl;
    }
}
```

timus/1258.cpp

```cpp
//1258
//Pool
//Math;Geometry;Mirror
#include <iostream>
#include <cmath>
#include <string>
#include <iomanip>
#define ull long long

using namespace std;

int main() {
```

```
    ull W, D, a, b, c, d;
    string s;
    while(cin >> W >> D >> a >> b >> c >> d) {
        cin >> s;
        ull x=0, y=0;
        ull sr=1<<30, sl=1<<30, sf=1<<30, sb=1<<30;

        for(ull i=0; i<s.size(); i++) {
            switch(s[i]) {
                case 'R': x+=2*(W-c); sr = min(sr, i); break;
                case 'L': x+=2*c; sl = min(sl, i); break;
                case 'F': y+=2*d; sf = min(sf, i); break;
                case 'B': y+=2*(D-d); sb = min(sb, i); break;
            }
        }

        c += sr<sl?x:-x;
        d += sb<sf?y:-y;

        cout << fixed << setprecision(4) << sqrt((a-c)*(a-c)+(b-d)*(b-d)+0.0) << endl;

    }
}
```

timus/1332.cpp

```
//1332
//Genie Bomber
//Math;Geometry;Enclosing Circle
#include <iostream>
#include <cmath>
#define EP 1e-6
using namespace std;

struct Point {
    double x, y;

    Point() {}
    Point(double x, double y) : x(x), y(y) {}

    double dist(Point A) {
        return sqrt(pow(A.x-x,2)+pow(A.y-y,2));
    }

    Point middle(Point B) {
        return Point((x-B.x)/2, (y-B.y)/2);
    }

};

struct Circle {
    Point c; double r;
    Circle(Point c, double r) : c(c), r(r) { }

    Circle(Point p1, Point p2, double r) {
        double d2 = (p1.x - p2.x) * (p1.x - p2.x) +
                    (p1.y - p2.y) * (p1.y - p2.y);
        double det = r*r / d2 - 0.25;
        double h = sqrt(det);

        this->c = Point((p1.x + p2.x) * 0.5 + (p1.y - p2.y) * h,
                        (p1.y + p2.y) * 0.5 + (p2.x - p1.x) * h);
        this->r = r;
    }


    static bool invalid(Point p1, Point p2, double r) {
        double d2 = (p1.x - p2.x) * (p1.x - p2.x) +
                    (p1.y - p2.y) * (p1.y - p2.y);
        double det = r*r / d2 - 0.25;
        return det < 0.0;
    }


    bool within(Point p) {
        return c.dist(p)-r < EP;
    }
};

Point P[106];

```

```
int best(Circle c1, int n) {
    int sum1 = 0;
    for(int k=0; k<n; k++) {
        if (c1.within(P[k]))
            sum1++;
    }
    return sum1;
}


int main() {
    int n, r, R;
    while(cin >> n) {
        for(int i=0; i<n; i++)
            cin >> P[i].x >> P[i].y;
        cin >> R >> r;
        R-=r;

        int maxx = 0;
        for(int i=0; i<n; i++) {
            for(int j=0; j<n; j++) {
                if (i==j) {
                    maxx = max(maxx, best(Circle(P[i], R), n));
                } else if (!Circle::invalid(P[i], P[j], R)) {
                    maxx = max(maxx, best(Circle(P[i], P[j], R), n));
                    maxx = max(maxx, best(Circle(P[i], P[j], R), n));
                }
            }
        }

        cout << maxx << endl;
    }
}
```

timus/1373.cpp

```
//1373
//Pictura ex Machina
//Math;Geometry;Segment Rotation
#include <iostream>
#include <cmath>
#include <iomanip>
using namespace std;

double PI = 2*acos(0.0);

struct Point {
    double x, y;

    Point() {}
    Point(double x, double y) : x(x), y(y) {}

    Point rotateWith(const Point origin, double si, double co, double scale) const {
        double tx = this->x - origin.x;
        double ty = this->y - origin.y;
        double x = (tx * co + ty * si)/scale;
        double y = (tx * -si + ty * co)/scale;
        return Point(origin.x + x, origin.y + y);

    }
};

double round4(double a) {
    if (a<0) return 0.0;
    return floor(a*10000+0.5)/10000.0;
}

int main() {
    Point a, b;
    double minx=1<<30, maxx=0, miny=1<<30, maxy=0;
    while(cin >> a.x >> a.y >> b.x >> b.y) {
        Point c = b.rotateWith(a, sin(-PI/4), cos(-PI/4), sqrt(2.0));

        minx = min(minx, min(min(a.x, b.x), c.x));
        miny = min(miny, min(min(a.y, b.y), c.y));
        maxx = max(maxx, max(max(a.x, b.x), c.x));
        maxy = max(maxy, max(max(a.y, b.y), c.y));
    }

    cout << fixed << setprecision(4) << round4(maxx - minx) << " " << round4(maxy - miny) << endl;
}
```

timus/1422.cpp

```cpp
//1422
//Fireflies
//Math;Geometry;3D Line Detection
#include <iostream>
#include <cmath>
#include <map>
using namespace std;

int gcd(int a, int b) {
    while(b)
        swap(a=a%b,b);
    return a;
}

struct Vector {
    int x, y, z;

    Vector() : x(0), y(0), z(0) {}
    Vector(int x, int y, int z) : x(x), y(y), z(z) {}


    Vector normalize() const {
        int d = gcd(x, gcd(y, z));
        return Vector(x/d, y/d, z/d);
    }

    Vector operator -(const Vector& that) const {
        return Vector(x-that.x, y-that.y, z-that.z);
    }

    bool operator <(const Vector& that) const {
       if (x!=that.x) return x<that.x;
       if (y!=that.y) return y<that.y;
       return z<that.z;
    }

};


Vector P[2007];
map<Vector, int> M;

int main() {
    int n;
    while(cin >> n) {
        for(int i=0; i<n; i++)
            cin >> P[i].x >> P[i].y >> P[i].z;

        int maxx = 0;
        for(int i=0; i<n; i++) {
            M.clear();
            for(int j=i+1; j<n; j++)
                maxx = max(maxx, ++M[(P[i]-P[j]).normalize()]);
        }

        cout << maxx+1 << endl;

    }
}
```

timus/1578.cpp

```cpp
//1578
//Mammoth Hunt
//Math;Geometry;Segments Angle
#include <iostream>
#include <cmath>
#define EP 1e-6
#define PI 3.14159265
using namespace std;

struct Point {
    int x, y;

    Point() {}
    Point(int x, int y) : x(x), y(y) {}

```

```
    double dist(Point A) {
        return sqrt(pow(A.x-x,2.0)+pow(A.y-y,2.0));
    }

    double angle(Point B, Point C) {
        double a = dist(B), b = B.dist(C), c=dist(C);
        double ret = acos((a*a+b*b-c*c) / (2*a*b));
        if (ret < 0) ret += 2*PI;
        //cout << " " << a << " " << b << " " << c << " " << ret << endl;
        return ret;
    }

    bool accute(Point B, Point C) {
        return angle(B, C) < PI/2.0;
    }

};

Point P[2010];
int O[2010], V[2010];
int n;

bool dfs(int v, int i) {
    O[i] = v; V[v] = i;
    if (i==n) return true;

    for(int j=1; j<=n; j++) {
        if (V[j]) continue;
        Point a = P[O[i-1]], b = P[O[i]], c = P[j];
        if (a.accute(b,c))
            if (dfs(j, i+1))
                return true;
    }
    V[v] = 0;

    return false;

}

int main() {
    int k;
    while(cin >> k) {
        memset(O, 0, sizeof(O));
        memset(V, 0, sizeof(V));

        n = k + 2;
        for(int i=1; i<=n; i++)
            cin >> P[i].x >> P[i].y;

        bool ok = false;
        for(int i=1; i<=n && !ok; i++) {
            O[1] = i; V[i] = 1;
            for(int j=1; j<=n && !ok; j++) {
                if (i==j) continue;
                ok |= dfs(j, 2);
            }
        }

        if (ok) {
            cout << "YES" << endl;
            for(int i=1; i<=n; i++) {
                if (i>1) cout << " ";
                cout << O[i];
            }
            cout << endl;
        } else {
            cout << "NO" << endl;
        }


    }
}
```

timus/1658.cpp

```
//1658
//Sum of Digits
//Dynamic Programming;Ad hoc
#include <iostream>
#include <vector>
#include <cstring>
```

```
using namespace std;

short S[1010][9000];
short T[1010][9000];

int main() {
    T[0][0] = 1;
    for(int i=1; i<=1000; i++) {
        for(int j=1; j<=8100; j++) {
            S[i][j] = 102;
            for(int k=1; k<=9; k++) {
                int a = i-k;
                int b = j-k*k;
                if (a>=0 && b>=0 && T[a][b] && S[a][b]+1<S[i][j]) {
                    T[i][j] = k;
                    S[i][j] = S[a][b]+1;
                }
            }
        }
    }

    int t; cin >> t;
    while(t--) {
        int s1, s2;
        cin >> s1 >> s2;

        if (s1 > 1000 || s2 > 8100 || S[s1][s2] > 100) {
            cout << "No solution" << endl;
            continue;
        }

        int n = S[s1][s2];
        for(int i=0; i<n; i++) {
            int d = T[s1][s2];
            cout << d;
            s1 -= d;
            s2 -= d*d;
        }
        cout << endl;

    }
}
```

spoj/nkmobile.cpp

```
//NKMOBILE
//IOI01 Mobiles
//Misc;Fenwick Tree;2D
#include <iostream>
#include <cstring>
#define MAX 1030
using namespace std;

struct Fenwick2D {
    int T[MAX][MAX];
    int n, m;

    void clear(int n, int m) {
        for(int i=1; i<=n; i++)
            for(int j=1; j<=m; j++)
                T[i][j] = 0;

        this->n = n;
        this->m = m;
    }

    void adjust(int x, int y, int v) {
        for (int i=x; i <= n; i += (i&-i))
            for(int j=y; j <= m; j += (j&-j))
                T[i][j] += v;
    }

    int rsq(int x, int y) {
        int sum = 0;
        for(int i=x; i; i -= (i&-i))
            for(int j=y; j; j -= (j&-j))
                sum += T[i][j];
        return sum;
    }

    int rsq(int x1, int y1, int x2, int y2) {
```

```
        return rsq(x2, y2) - rsq(x2, y1-1) - rsq(x1-1, y2) + rsq(x1-1, y1-1);
    }
};

Fenwick2D T;

int main() {
    int cmd;
    while(cin >> cmd, cmd != 3) {
        if (cmd == 0) {
            int s;
            cin >> s;
            T.clear(s, s);
        } else if (cmd == 1) {
            int x, y, a;
            cin >> x >> y >> a;
            x++; y++;
            T.adjust(x, y, a);
        } else if (cmd == 2) {
            int x1, y1, x2, y2;
            cin >> x1 >> y1 >> x2 >> y2;
            x1++; y1++; x2++; y2++;
            cout << T.rsq(x1, y1, x2, y2) << endl;
        }
    }
}
```

spojbr/homem.cpp

```
//HOMEM
//Homem, Elefante e Rato
//Misc;Segment Tree;Lazy Propagation
#include <iostream>
#include <cstring>
#include <cstdio>
#define MAX 600100
#define ull long long
using namespace std;

struct Node {
    int a, b, c;
    int pending;

    Node() {}
    Node(int a) : a(a), b(0), c(0), pending(0) { }
    Node(int a, int b, int c) : a(a), b(b), c(c), pending(0) {}

    Node change(int n) {
        n%=3;
        pending += n;
        if (n==1) {
            swap(a, b); swap(a, c);
        } else if (n==2) {
            swap(c, b); swap(c, a);
        }
        return *this;
    }

    Node operator +(Node x) {
        return Node(a+x.a, b+x.b, c+x.c);
    }
};

struct Segtree {
    Node T[MAX];
    int n;

    Segtree() {
        clear(1);
    }

    void clear(int n) {
        while(n != n&-n)
            n += n&-n;

        this->n = n;

        build(1, 1, n);
    }

    void build(int v, int a, int b) {
```

```
        T[v] = Node(b-a+1);

        if (a>=b) return;
        build(2*v, a, (a+b)/2);
        build(2*v+1, (a+b)/2+1, b);
    }

    Node update(int v, int a, int b, int i, int j, int carry, int increment) {
        T[v].change(carry);

        if (i>b || j<a)
            return Node(0);

        if (i<=a && b<=j)
            return T[v].change(increment);

        Node answer =
            update(v*2, a, (a+b)/2, i, j, T[v].pending, increment) +
            update(v*2+1, (a+b)/2+1, b, i, j, T[v].pending, increment);

        T[v] = T[v*2] + T[v*2+1];

        return answer;
    }

    Node update(int i, int j, int inc) {
        return update(1, 1, n, i, j, 0, inc);
    }

    Node query(int i, int j) {
        return update(1, 1, n, i, j, 0, 0);
    }

};

Segtree T;

int main() {
    int n, m;
    while(scanf("%d%d", &n, &m) != EOF) {
        T.clear(n);
        for(int i=0; i<m; i++) {
            char cmd; int a, b;
            scanf(" %c %d %d", &cmd, &a, &b);
            if (cmd == 'M') {
                T.update(a, b, 1);
            } else {
                Node node = T.query(a, b);
                printf("%d %d %d\n", node.a, node.b, node.c);
            }
        }
        printf("\n");
    }
}
```