

# 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 - Hippiity 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 Attacks!
  - **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

```
1 //109
2 //SCUD Busters
3 //Math;Geometry;Convex Hull
4 #include <iostream>
5 #include <cmath>
6 #include <iomanip>
7 #include <algorithm>
8 using namespace std;
9
10 struct Point {
11     int x, y;
12
13     Point() {}
14     Point(int x, int y) : x(x), y(y) {}
15
16     bool left(Point& a, Point& b) {
17         return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x) < 0;
18     }
19
20     static bool lesserX(Point& p1, Point& p2) {
21         return p1.x < p2.x;
22     }
23
24     bool operator ==(const Point& p) const {
25         return this->x == p.x && this->y == p.y;
26     }
27 };
28
29 double area(Point* A, int a) {
30     double area = 0;
31     for(int i=0; i<a; i++) {
32         int j = (i+1)%a;
33         area += (A[i].x + A[j].x) * (A[i].y - A[j].y);
34     }
35     return area / 2;
36 }
37
38 int convexHull(Point* P, int n, Point* S) {
39     int m=0;
40     S[m++] = *min_element(P, P+n, Point::lesserX);
41     while(true) {
42         Point cand = S[m-1];
43         for(int j=0; j<n; j++)
44             if (cand==S[m-1] or P[j].left(S[m-1], cand))
45                 cand = P[j];
46
47         if (cand == S[0]) break;
48         S[m++] = cand;
49     }
50     return m;
51 }
52
53 //assumes convex polygon, ordered clockwise
54 //convex hull output is just that
55 bool checkInside(Point* P, int n, Point v) {
56     for(int i=1; i<=n; i++)
57         if (v.left(P[i-1], P[i%n]))
58             return false;
59
60     return true;
61 }
62
63 Point P[30][250], S[30][250];
64 int N[30], M[30];
65 bool V[30];
66
67 int main() {
68     int n, k=0;
69     while(cin >> n, n!=-1) {
70         N[k] = n;
71         for(int i=0; i<n; i++) {
72             int x,y; cin >> x >> y;
73             P[k][i] = Point(x,y);
74         }
75
76         M[k] = convexHull(P[k], n, S[k]);
77
78         k++;
79     }
80 }
```

```

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) {}
12     inline const bool operator < (const Number& that) const {
13         return n > that.n;
14     }
15 };

```



```

16 priority_queue<Number> Q;
17
18 int main() {
19     Q.push(Number(1,1));
20     for(int i=1; i<1500; i++) {
21         Number last = Q.top(); Q.pop();
22         if (last.p <= 2) Q.push(Number(last.n*2, 2));
23         if (last.p <= 3) Q.push(Number(last.n*3, 3));
24         if (last.p <= 5) Q.push(Number(last.n*5, 5));
25     }
26
27     cout << "The 1500'th ugly number is " << Q.top().n << "." << endl;
28
29     return 0;
30 }

```

## uva/160.cpp

```

1 //160
2 //Factors and Factorials
3 //Misc;Ad hoc
4 #include <iostream>
5 #include <iomanip>
6 using namespace std;
7
8 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 = 24;
9 int T[101][25];
10
11 int main() {
12     for(int i=2; i<=100; i++) {
13         int p = i;
14         for(int j=0; j<wn; j++)
15             T[i][j] = T[i-1][j];
16
17         for(int j=0; j<wn && p>1; j++) {
18             while(p%W[j]==0) {
19                 p/=W[j];
20                 T[i][j]++;
21             }
22         }
23     }
24
25     int n;
26     while(cin >> n, n) {
27         cout << setw(3) << right << n << "! =";
28         int count = 0;
29         for(int i=0; i<wn; i++) {
30             if (T[n][i] == 0) break;
31             if (++count > 15) {
32                 count = 1;
33                 cout << endl << " ";
34             }
35             cout << setw(3) << right << T[n][i];
36         }
37         cout << endl;
38     }
39
40     return 0;
41 }

```

## uva/245.cpp

```

1 //245
2 //Uncompress
3 //Misc;Linked List
4 #include <iostream>
5 #include <sstream>
6 #include <string>
7 #include <climits>
8 #include <cstring>
9 #include <cstdio>
10 #include <list>
11 #define MAX 1000
12 using namespace std;
13
14 list<string> W;
15 stringstream sin;
16 int curnum=0;
17 bool word=false, number=false;
18

```

```

19 void finishWord() {
20     W.push_back(sin.str());
21     sin.str("");
22     word = false;
23 }
24
25 void finishNumber() {
26     list<string>::iterator it = W.end();
27     while(curnum-->0)
28         it--;
29
30     cout << *it;
31     W.push_back(*it);
32     W.erase(it);
33
34     curnum = 0;
35     number = false;
36 }
37
38 int main() {
39     string s;
40
41     while(getline(cin, s), s!="0") {
42         for(int i=0; i<s.size(); i++) {
43             char c = s[i];
44             if (c >= 'a' && c <= 'z' || c >= 'A' && c <= 'Z') {
45                 sin << c;
46                 word = true;
47             } else if (word) finishWord();
48
49             if (c >= '0' && c <= '9') {
50                 curnum *= 10; curnum += c-'0';
51                 number = true;
52             } else if (number) finishNumber();
53
54             if (!number)
55                 cout << c;
56         }
57         if (word) finishWord();
58         if (number) finishNumber();
59
60         cout << endl;
61     }
62 }

```

## uva/273.cpp

```

1 //273
2 //Jack Straws
3 //Graphs;DFS
4 #include <iostream>
5 #include <cstring>
6 #define MAX 100002
7 using namespace std;
8
9 static bool segment(int xi, int yi, int xj, int yj,
10                    int xk, int yk) {
11     return (xi <= xk || xj <= xk) && (xk <= xi || xk <= xj) &&
12            (yi <= yk || yj <= yk) && (yk <= yi || yk <= yj);
13 }
14
15 static char direction(int xi, int yi, int xj, int yj,
16                      int xk, int yk) {
17     int a = (xk - xi) * (yj - yi);
18     int b = (xj - xi) * (yk - yi);
19     return a < b ? -1 : a > b ? 1 : 0;
20 }
21
22 bool intersect(int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {
23     char d1 = direction(x3, y3, x4, y4, x1, y1);
24     char d2 = direction(x3, y3, x4, y4, x2, y2);
25     char d3 = direction(x1, y1, x2, y2, x3, y3);
26     char d4 = direction(x1, y1, x2, y2, x4, y4);
27     return (((d1 > 0 && d2 < 0) || (d1 < 0 && d2 > 0)) &&
28            ((d3 > 0 && d4 < 0) || (d3 < 0 && d4 > 0))) ||
29            (d1 == 0 && segment(x3, y3, x4, y4, x1, y1)) ||
30            (d2 == 0 && segment(x3, y3, x4, y4, x2, y2)) ||
31            (d3 == 0 && segment(x1, y1, x2, y2, x3, y3)) ||
32            (d4 == 0 && segment(x1, y1, x2, y2, x4, y4));
33 }
34
35 int G[20][20], V[20], A[20], B[20], C[20], D[20], n;
36

```

```

37 int dfs(int v, int comp) {
38     V[v] = comp;
39     for(int i=1; i<=n; i++)
40         if (!V[i] && G[v][i])
41             dfs(i, comp);
42 }
43
44 int main() {
45     int t; cin >> t; t=0;
46     while(cin >> n) {
47         memset(G, 0, sizeof(G));
48         memset(V, 0, sizeof(V));
49         for(int i=1; i<=n; i++) {
50             cin >> A[i] >> B[i] >> C[i] >> D[i];
51             for(int j=1; j<i; j++)
52                 G[i][j] = G[j][i] = intersect(A[i], B[i], C[i], D[i], A[j], B[j], C[j], D[j]);
53         }
54
55         int compn = 0;
56         for(int i=1; i<=n; i++)
57             if (!V[i])
58                 dfs(i, ++compn);
59
60         if (t++) cout << endl;
61         int a, b;
62         while(cin >> a >> b, a|b) {
63             cout << (V[a] == V[b]?"CONNECTED":"NOT CONNECTED") << endl;
64         }
65     }
66     return 0;
67 }

```

## uva/298.cpp

```

1 //298
2 //Race Tracks
3 //Graphs;Shortest Path;BFS
4 #include <iostream>
5 #include <cstring>
6 #include <queue>
7 #define MAX 30
8 using namespace std;
9
10 bool V[MAX][MAX][7][7];
11 int X, Y;
12
13 struct Step {
14     int x, y, a, b, v;
15     Step() {}
16     Step(int x, int y, int a, int b, int v) : x(x), y(y), a(a), b(b), v(v) {}
17
18     bool valid() {
19         return x>=0 && x<X && y>=0 && y<Y && a >= -3 && a <= 3 && b >= -3 && b <= 3 && !V[x][y][a+3][b+3];
20     }
21
22     void mark() {
23         V[x][y][a+3][b+3] = true;
24     }
25
26     Step go(int mx, int my) {
27         return Step(x+a+mx, y+b+my, a+mx, b+my, v+1);
28     }
29 };
30
31 int main() {
32     int t; cin >> t; t=0;
33     while(cin >> X >> Y) {
34         memset(V, 0, sizeof(V));
35         int x1, y1, x2, y2;
36         cin >> x1 >> y1 >> x2 >> y2;
37
38         int p, px1, px2, py1, py2;
39         cin >> p;
40         while(p--) {
41             cin >> px1 >> px2 >> py1 >> py2;
42             for(int i=px1; i<=px2; i++)
43                 for(int j=py1; j<=py2; j++)
44                     for(int ai=0; ai<=6; ai++)
45                         for(int bi=0; bi<=6; bi++)
46                             V[i][j][ai][bi] = true;
47         }
48
49         bool found = false;

```

```

50     queue<Step> Q;
51     Q.push(Step(x1, y1, 0, 0, 0));
52
53     while(!Q.empty()) {
54         Step it = Q.front(); Q.pop();
55         if (!it.valid()) continue;
56         it.mark();
57
58         if (it.x == x2 && it.y == y2) {
59             cout << "Optimal solution takes " << it.v << " hops." << endl;
60             found = true;
61             break;
62         }
63
64         for(int ai=-1; ai<=1; ai++)
65             for(int bi=-1; bi<=1; bi++)
66                 Q.push(it.go(ai, bi));
67     }
68     if (!found) cout << "No solution." << endl;
69 }
70 }

```

## uva/314.cpp

```

1  //314
2  //Robot
3  //Graphs;Shortest Path;BFS
4  #include <iostream>
5  #include <iomanip>
6  #include <cstring>
7  #include <string>
8  #include <cmath>
9  #include <climits>
10 #include <vector>
11 #define MAX 70
12 using namespace std;
13
14 int G[MAX][MAX], n, m, sx, sy, tx, ty;
15 bool V[MAX][MAX][4];
16 string dir;
17
18 struct Step {
19     int x, y, d, v, p;
20     Step() {}
21     Step(int x, int y, int d, int v, int p) : x(x), y(y), d(d), v(v), p(p) {}
22     Step left(int pp) {
23         return Step(x, y, (d+3)%4, v+1, pp);
24     }
25     Step right(int pp) {
26         return Step(x, y, (d+1)%4, v+1, pp);
27     }
28     bool canGo(int i) {
29         return (d==0 && x-i>=1 && !G[x-i][y]) ||
30                (d==1 && y+i<m-1 && !G[x][y+i]) ||
31                (d==2 && x+i<n-1 && !G[x+i][y]) ||
32                (d==3 && y-i>=1 && !G[x][y-i]);
33     }
34     Step go(int pp, int i) {
35         if (d==0) return Step(x-i, y, d, v+1, pp);
36         if (d==1) return Step(x, y+i, d, v+1, pp);
37         if (d==2) return Step(x+i, y, d, v+1, pp);
38         if (d==3) return Step(x, y-i, d, v+1, pp);
39     }
40 };
41
42 int main() {
43     while(cin >> n >> m, n|m) {
44         vector<Step> Q;
45
46         memset(G, 0, sizeof(G));
47         memset(V, 0, sizeof(V));
48
49         for(int i=0; i<n; i++)
50             for (int j=0; j<m; j++)
51                 cin >> G[i][j];
52
53         n++; m++;
54         for(int i=n-1; i>=0; i--)
55             for (int j=m-1; j>=0; j--)
56                 if (G[i][j])
57                     G[i+1][j] = G[i][j+1] = G[i+1][j+1] = 1;
58     }
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;
41             }
42         }
43         dfs(1, 1); P[1]--;
44         int cnt = 0;
45         for(int i=1; i<=n; i++)
46             if (P[i]) cnt++;
47
48         cout << cnt << endl;
49     }
50 }

```

## uva/321.cpp

```
1 //321
2 //The New Villa
3 //Graphs;Shortest Path;BFS
4 #include <iostream>
5 #include <cstring>
6 #include <climits>
7 #include <vector>
8 #define MAX 15
9 using namespace std;
10
11
12
13 int G[MAX][MAX], C[MAX][MAX], n, m1, m2;
14 bool V[MAX][1200];
15 string dir;
16 struct Step {
17     int x, s, v, p;
18     int type, room;
19     Step() {}
20     Step(int x, int s, int v, int p) : x(x), s(s), v(v), p(p) {}
21     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) {}
22
23     Step change(int pp, int i) {
24         return Step(x, s ^ (1<<i), v+1, pp, (s & (1<<i))?2:1, i);
25     }
26     Step move(int pp, int i) {
27         return Step(i, s, v+1, pp, 3, i);
28     }
29 };
30
31 vector<Step> Q;
32
33 void print(Step step) {
34     if (step.p == -1) return;
35     print(Q[step.p]);
36     if (step.type == 1)
37         cout << "- Switch on light in room " << step.room+1 << "." << endl;
38     if (step.type == 2)
39         cout << "- Switch off light in room " << step.room+1 << "." << endl;
40     if (step.type == 3)
41         cout << "- Move to room " << step.room+1 << "." << endl;
42 }
43
44
45 int main() {
46     int tt=0;
47     while(cin >> n >> m1 >> m2, n|m1|m2) {
48         Q = vector<Step>();
49
50         memset(G, 0, sizeof(G));
51         memset(C, 0, sizeof(C));
52         memset(V, 0, sizeof(V));
53
54         int a, b;
55         for(int i=0;i<m1; i++) {
56             cin >> a >> b;
57             a--; b--;
58             G[a][b] = G[b][a] = 1;
59         }
60         for(int i=0;i<m2; i++) {
61             cin >> a >> b;
62             a--;b--;
63             C[a][b] = 1;
64         }
65
66         Q.push_back(Step(0, 1, 0, -1));
67
68         int ptr = 0;
69         cout << "Villa #" << ++tt << endl;
70         while(ptr < Q.size()) {
71             Step it = Q[ptr];
72             if (it.x == n-1 && it.s == (1<<(n-1))) {
73                 cout << "The problem can be solved in " << it.v << " steps:" << endl;
74                 print(it);
75                 break;
76             }
77
78             if (V[it.x][it.s]) { ptr++; continue; }
79             V[it.x][it.s] = true;
80
81
```

```

82         for(int i=0; i<n; i++) {
83             if (G[it.x][i] && (it.s & (1<<i))) Q.push_back(it.move(ptr, i));
84             if (C[it.x][i] && it.x != i) Q.push_back(it.change(ptr, i));
85         }
86         ptr++;
87     }
88     if (ptr == Q.size()) cout << "The problem cannot be solved." << endl;
89     cout << endl;
90 }
91 }
92 }

```

## uva/361.cpp

```

1  //361
2  //Cops and Robbers
3  //Math;Geometry;Convex Hull;Monotone Chain
4  #include <iostream>
5  #include <cmath>
6  #include <iomanip>
7  #include <algorithm>
8  using namespace std;
9
10 struct Point {
11     int x, y;
12
13     Point() {}
14     Point(int x, int y) : x(x), y(y) {}
15
16     int product(Point a, Point b) {
17         return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
18     }
19
20     bool left(Point a, Point b) {
21         return product(a, b) < 0;
22     }
23
24     bool operator <(const Point& p) const {
25         if (this->x != p.x) return this->x < p.x;
26         return this->y < p.y;
27     }
28
29     bool operator ==(const Point& p) const {
30         return this->x == p.x and this->y == p.y;
31     }
32
33     bool insideSegment(Point a, Point b) {
34         return product(a, b) == 0 && min(a,b) < *this && *this < max(a,b);
35     }
36 };
37
38 int convexHull(Point* P, int n, Point* S) {
39     sort(P, P+n);
40
41     int m=0;
42     for(int i=0; i<n; i++) {
43         while(m >= 2 && S[m-1].left(S[m-2], P[i])) m--;
44         S[m++] = P[i];
45     }
46     m--;
47
48     for(int i=n-1, k=m; i >= 0; i--) {
49         while(m >= k+2 && S[m-1].left(S[m-2], P[i])) m--;
50         S[m++] = P[i];
51     }
52     m--;
53
54     return m;
55 }
56
57 bool checkInside(Point* P, int n, Point v) {
58     for(int i=0; i<n; i++) {
59         int j = (i+1)%n;
60         if (v == P[i] || v.insideSegment(P[i], P[j]))
61             return true;
62     }
63
64     for(int i=0; i<n; i++) {
65         int j = (i+1)%n;
66         if (!v.left(P[i], P[j]))
67             return false;
68     }
69 }

```

```

70     return true;
71 }
72
73
74 Point C[250], R[250], CS[250], RS[250];
75
76 int main() {
77     int c, r, o, tt=1;
78     while(cin >> c >> r >> o, c|r|o) {
79         for(int i=0; i<c; i++)
80             cin >> C[i].x >> C[i].y;
81
82         for(int i=0; i<r; i++)
83             cin >> R[i].x >> R[i].y;
84
85         int cs = convexHull(C, c, CS);
86         int rs = convexHull(R, r, RS);
87
88         /*for(int i=0; i<cs; i++) {
89             cout << " " << CS[i].x << " " << CS[i].y << endl;
90         }*/
91
92         cout << "Data set " << tt++ << ":" << endl;
93         for(int i=0; i<o; i++) {
94             Point p; cin >> p.x >> p.y;
95             cout << "      Citizen at (" << p.x << ", " << p.y << ") is ";
96             if (checkInside(CS, cs, p) && cs > 2)
97                 cout << "safe." << endl;
98             else if (checkInside(RS, rs, p) && rs > 2)
99                 cout << "robbed." << endl;
100             else
101                 cout << "neither." << endl;
102         }
103         cout << endl;
104     }
105 }
106 }

```

## uva/422.cpp

```

1  //422
2  //Word-Search Wonder
3  //Misc;String Matching;KMP;2D
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #define MAX 105
8  using namespace std;
9
10 char C[MAX][MAX];
11 int F[MAX];
12 int n;
13
14 void kmp_init(string& P) {
15     F[0] = 0; F[1] = 0;
16     int i = 1, j = 0;
17     while(i<P.size()) {
18         if (P[i] == P[j])
19             F[++i] = ++j;
20         else if (j == 0)
21             F[++i] = 0;
22         else
23             j = F[j];
24     }
25 }
26
27 bool kmp(string& P, int x, int y, int mx, int my) {
28     kmp_init(P);
29     int j = 0, m = P.size();
30
31     while(x >= 0 && x < n && y >= 0 && y < n) {
32         while(j < m) {
33             if (P[j] == C[x][y]) {
34                 x+=mx; y+=my; j++;
35             } else break;
36         }
37         if (j == m) {
38             cout << x-m*mx+1 << ", " << y-m*my+1 << " " << x+1-mx << ", " << y+1-my << endl;
39             return true;
40         }
41         else if (j == 0) { x+=mx; y+=my; };
42         j = F[j];
43     }

```



```

44     return false;
45 }
46
47 int main() {
48     cin >> n;
49     for(int i=0; i<n; i++)
50         for(int j=0; j<n; j++)
51             cin >> C[i][j];
52
53     string P;
54     while(cin >> P, P!="0") {
55         bool result = false;
56         for(int i=0; i<n; i++) {
57             result = result || kmp(P, i, 0, 0, 1);
58             result = result || kmp(P, i, n-1, 0, -1);
59             result = result || kmp(P, 0, i, 1, 0);
60             result = result || kmp(P, n-1, i, -1, 0);
61
62             result = result || kmp(P, 0, i, 1, 1);
63             result = result || kmp(P, i, n-1, -1, -1);
64             result = result || kmp(P, i, 0, 1, 1);
65             result = result || kmp(P, n-1, i, -1, -1);
66
67             result = result || kmp(P, 0, i, -1, 1);
68             result = result || kmp(P, i, 0, 1, -1);
69             result = result || kmp(P, i, n-1, 1, -1);
70             result = result || kmp(P, n-1, i, -1, 1);
71         }
72
73         if (!result)
74             cout << "Not found" << endl;
75     }
76 }

```

## uva/424.cpp

```

1  //424
2  //Integer Inquiry
3  //Math;Big Integer
4  #define MAX 110
5  #include <iostream>
6  #include <string>
7  using namespace std;
8
9  int T[MAX];
10 int n=0;
11
12 void add(string& s) {
13     int c=0;
14     int m = s.size();
15     for(int i=0; i<m; i++) {
16         int a = s[m-i-1]-'0';
17         T[i] += a+c;
18         c = T[i]/10;
19         T[i] %= 10;
20     }
21     n = max(n, m);
22     while (c) {
23         T[m] += c;
24         c = T[m]/10;
25         T[m] %= 10;
26         n = max(n, ++m);
27     }
28 }
29
30
31 int main() {
32     string s;
33     while(cin >> s, s!="0")
34         add(s);
35
36     for(int i=n-1; i>=0; i--)
37         cout << T[i];
38     cout << endl;
39 }

```

## uva/458.cpp

```

1  //458
2  //The Decoder
3  //Misc;Ad hoc

```

```

4 | #include <iostream>
5 | #include <string>
6 | using namespace std;
7 |
8 | int main() {
9 |     for(string s; cin>>s;) {
10 |         for(int i=0; i<s.size(); i++)
11 |             s[i]-=7;
12 |         cout << s << endl;
13 |     }
14 | }

```

## uva/473.cpp

```

1 | //473
2 | //Raucous Rockers
3 | //Dynamic Programming;Longest Increasing Subsequence
4 | #include <iostream>
5 | #include <string>
6 | #include <cstring>
7 | #include <cmath>
8 | #define MAX 10005
9 | using namespace std;
10 |
11 | int S[MAX], T[MAX];
12 | char skip;
13 | int main() {
14 |     int n,t,m,cases;
15 |     cin >> cases;
16 |     while(cases--) {
17 |         cin >> n >> t >> m;
18 |         memset(T, 0x3F, sizeof(T));
19 |
20 |         for(int i=1; i<=n; i++) {
21 |             cin >> S[i];
22 |             if (i<n) cin >> skip;
23 |         }
24 |
25 |         int k=0;
26 |         T[0] = 0;
27 |
28 |         for(int i=1;i<=n;i++) {
29 |             for(int j=k; j>=0; j--) {
30 |                 int add = 0;
31 |                 if ((T[j]%t)+S[i] > t) add = t-T[j]%t;
32 |                 if (T[j]+S[i]+add <= T[j+1]) {
33 |                     T[j+1] = T[j]+S[i]+add;
34 |                     k=max(k, j+1);
35 |                 }
36 |             }
37 |         }
38 |
39 |         int answer = 0;
40 |         for(int i=0; i<=k && T[i] <= m*t; i++)
41 |             answer = i;
42 |
43 |         cout << answer << endl;
44 |         if (cases) cout << endl;
45 |     }
46 |
47 |     return 0;
48 | }

```

## uva/494.cpp

```

1 | //494
2 | //Kindergarten Counting Game
3 | //Misc;Ad hoc
4 | #include <iostream>
5 | #include <string>
6 | using namespace std;
7 |
8 | int main() {
9 |     string s;
10 |     while(getline(cin, s)) {
11 |         bool inside = false;
12 |         int words = 0;
13 |         for(int i=0; i<s.size(); i++) {
14 |             if (s[i] >= 'a' && s[i] <= 'z' || s[i] >= 'A' && s[i] <= 'Z') {
15 |                 inside = true;
16 |             } else if (inside) {

```

```

17         inside = false;
18         words++;
19     }
20 }
21 if (inside) words++;
22 cout << words << endl;
23 }
24
25 }

```

## uva/573.cpp

```

1  //573
2  //The Snail
3  //Misc;Ad hoc
4  #include <iostream>
5  using namespace std;
6
7  int main() {
8      int h,u,d,f;
9      while(cin >> h >> u >> d >> f, h|u|d|f) {
10         double current = 0, speed=u;
11
12         for (int i=1;;i++) {
13             current += speed;
14             speed = max(0.0, speed-f/100.0*u);
15             if (current > h) {
16                 cout << "success on day " << i << endl;
17                 break;
18             }
19             current -= d;
20             if (current < 0) {
21                 cout << "failure on day " << i << endl;
22                 break;
23             }
24         }
25     }
26
27     return 0;
28 }

```

## uva/579.cpp

```

1  //579
2  //ClockHands
3  //Misc;Ad hoc
4  #include <iostream>
5  #include <iomanip>
6  #include <cmath>
7  using namespace std;
8
9  int main() {
10     int x, y; char c;
11     while(cin >> x >> c >> y, x|y) {
12         double a = x*30+(y/2.0);
13         double b = y*6.0;
14         double r = abs(a-b);
15         if (r > 180.0)
16             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
39         for(int i=0; i<m; i++) {
40             int a, b; cin >> a >> b;
41             G[a][b] = G[b][a] = 1;
42         }
43
44         for(int i=1; i<=n; i++)
45             if (!V[i])
46                 dfs(i, i);
47
48         for(int i=1; i<=n; i++)
49             for(int j=1; j<=n; j++)
50                 if (G[i][j])
51                     cout << i << " " << j << endl;
52
53         cout << "#" << endl;
54     }
55 }

```

## uva/627.cpp

```

1  //627
2  //The Net

```

```

3 //Graphs;Shortest Path;BFS
4 #include <iostream>
5 #include <cstring>
6 #include <climits>
7 #include <vector>
8 #define MAX 400
9 using namespace std;
10
11 int G[MAX][MAX], n, m;
12 bool V[MAX];
13
14
15 struct Step {
16     int x, v, p;
17     Step() {}
18     Step(int x, int v, int p) : x(x), v(v), p(p) {}
19 };
20
21 vector<Step> Q;
22
23 void print(Step step, bool first) {
24     if (step.p != -1) print(Q[step.p], false);
25     cout << step.x << (first?" ":" ");
26 }
27
28 int main() {
29     while(cin >> n) {
30         cout << "-----" << endl;
31         memset(G, 0, sizeof(G));
32
33         int a, b;
34         for(int i=0; i<n;i++) {
35             cin >> a;
36             while(cin.get()!='\n') {
37                 if (cin.peek() == '\n') break;
38                 cin >> b;
39                 G[a][b] = true;
40             }
41         }
42
43         cin >> m;
44         for(int i=0;i<m;i++) {
45             memset(V, 0, sizeof(V));
46             cin >> a >> b;
47             Q = vector<Step>();
48             Q.push_back(Step(a, 0, -1));
49
50             int ptr = 0;
51             while(ptr < Q.size()) {
52                 Step it = Q[ptr];
53                 if (it.x == b) {
54                     print(it, true);
55                     cout << endl;
56                     break;
57                 }
58
59                 if (V[it.x]) { ptr++; continue; }
60                 V[it.x] = true;
61
62                 for(int i=1; i<=n; i++)
63                     if (G[it.x][i]) Q.push_back(Step(i,it.v+1,ptr));
64
65                 ptr++;
66             }
67             if (ptr == Q.size()) cout << "connection impossible" << endl;
68         }
69     }
70 }

```

## uva/652.cpp

```

1 //652
2 //Eight
3 //Graphs;Shortest Path;BFS
4 #include <iostream>
5 #include <map>
6 #include <string>
7 #include <sstream>
8 using namespace std;
9
10 struct Item {
11     int p, x;
12     string v;

```

```

13     char m;
14
15     Item() { }
16     Item(int p, string v, int x, char m) : p(p), v(v), x(x), m(m) { }
17 };
18
19 map<string, int> M;
20 Item Q[400000];
21 int qq=0;
22
23 void add(int p, string v, int i, int j, char m) {
24     if (not (i%3 == j%3 ^ i/3 == j/3) || j<0 || j>=9) return;
25     swap(v[i], v[j]);
26     if (M.find(v) != M.end()) return;
27     Q[M[v]=qq++] = Item(p, v, j, m);
28 }
29
30 int main() {
31     Q[M["123456780"]=qq++] = Item(-1, "123456780", 8, 'z');
32
33     for(int i=0; i<qq; i++) {
34         Item p = Q[i];
35
36         add(i, p.v, p.x, p.x-1, 'r');
37         add(i, p.v, p.x, p.x+1, 'l');
38         add(i, p.v, p.x, p.x-3, 'd');
39         add(i, p.v, p.x, p.x+3, 'u');
40     }
41
42
43     int tt; cin >> tt;
44     while(tt--) {
45         stringstream ss;
46         for(int i=0; i<9; i++) {
47             string s; cin >> s;
48             if (s=="x") s="0";
49             ss << s;
50         }
51
52         string s = ss.str();
53         if (M.find(s) == M.end()) {
54             cout << "unsolvable" << endl;
55         } else {
56             Item item = Q[M[s]];
57
58             while(item.p != -1) {
59                 cout << item.m;
60                 item = Q[item.p];
61             }
62             cout << endl;
63         }
64         if (tt) cout << endl;
65     }
66 }

```

## uva/719.cpp

```

1 //719
2 //Glass Beads
3 //Misc;String Matching;Suffix Array;Circular
4 #include <iostream>
5 #include <iomanip>
6 #include <cstring>
7 #include <string>
8 #include <cmath>
9 #define MAX 10050
10 using namespace std;
11
12 int RA[MAX], tempRA[MAX];
13 int SA[MAX], tempSA[MAX];
14 int C[MAX];
15
16 void suffix_sort(int n, int k) {
17     memset(C, 0, sizeof C);
18
19     for (int i = 0; i < n; i++)
20         C[RA[(i + k)%n]]++;
21
22     int sum = 0;
23     for (int i = 0; i < max(256, n); i++) {
24         int t = C[i];
25         C[i] = sum;
26         sum += t;
27     }
28 }

```

```

27     }
28
29     for (int i = 0; i < n; i++)
30         tempSA[C[RA[(SA[i] + k)%n]]++] = SA[i];
31
32     memcpy(SA, tempSA, n*sizeof(int));
33 }
34
35 void suffix_array(string &s) {
36     int n = s.size();
37
38     for (int i = 0; i < n; i++)
39         RA[i] = s[i];
40
41     for (int i = 0; i < n; i++)
42         SA[i] = i;
43
44     for (int k = 1; k < n; k *= 2) {
45         suffix_sort(n, k);
46         suffix_sort(n, 0);
47
48         int r = tempRA[SA[0]] = 0;
49         for (int i = 1; i < n; i++) {
50             int s1 = SA[i], s2 = SA[i-1];
51             bool equal = true;
52             equal &= RA[s1] == RA[s2];
53             equal &= RA[(s1+k)%n] == RA[(s2+k)%n];
54
55             tempRA[SA[i]] = equal ? r : ++r;
56         }
57
58         memcpy(RA, tempRA, n*sizeof(int));
59     }
60 }
61
62 int main() {
63     int tt; cin >> tt;
64     while(tt--) {
65         string s; cin >> s;
66         suffix_array(s);
67         cout << SA[0]+1 << endl;
68     }
69 }

```

## uva/760.cpp

```

1  //760
2  //DNA Sequencing
3  //Misc;String Matching;Suffix Array;Longest Common Prefix
4  #include <iostream>
5  #include <iomanip>
6  #include <cstring>
7  #include <string>
8  #include <cmath>
9  #define MAX 10050
10 using namespace std;
11
12 int RA[MAX], tempRA[MAX];
13 int SA[MAX], tempSA[MAX];
14 int C[MAX];
15 int Phi[MAX], PLCP[MAX], LCP[MAX];
16
17 void suffix_sort(int n, int k) {
18     memset(C, 0, sizeof C);
19
20     for (int i = 0; i < n; i++)
21         C[i + k < n ? RA[i + k] : 0]++;
22
23     int sum = 0;
24     for (int i = 0; i < max(256, n); i++) {
25         int t = C[i];
26         C[i] = sum;
27         sum += t;
28     }
29
30     for (int i = 0; i < n; i++)
31         tempSA[C[SA[i] + k < n ? RA[SA[i] + k] : 0]++] = SA[i];
32
33     memcpy(SA, tempSA, n*sizeof(int));
34 }
35
36 void suffix_array(string &s) {
37     int n = s.size();

```

```

38
39     for (int i = 0; i < n; i++)
40         RA[i] = s[i] - 1;
41
42     for (int i = 0; i < n; i++)
43         SA[i] = i;
44
45     for (int k = 1; k < n; k *= 2) {
46         suffix_sort(n, k);
47         suffix_sort(n, 0);
48
49         int r = tempRA[SA[0]] = 0;
50         for (int i = 1; i < n; i++) {
51             int s1 = SA[i], s2 = SA[i-1];
52             bool equal = true;
53             equal &= RA[s1] == RA[s2];
54             equal &= RA[s1+k] == RA[s2+k];
55
56             tempRA[SA[i]] = equal ? r : ++r;
57         }
58
59         memcpy(RA, tempRA, n*sizeof(int));
60     }
61 }
62
63 void lcp(string &s) {
64     int n = s.size();
65
66     Phi[SA[0]] = -1;
67     for (int i = 1; i < n; i++)
68         Phi[SA[i]] = SA[i-1];
69
70     int L = 0;
71     for (int i = 0; i < n; i++) {
72         if (Phi[i] == -1) {
73             PLCP[i] = 0;
74             continue;
75         }
76         while (s[i + L] == s[Phi[i] + L])
77             L++;
78
79         PLCP[i] = L;
80         L = max(L-1, 0);
81     }
82
83     for (int i = 1; i < n; i++)
84         LCP[i] = PLCP[SA[i]];
85 }
86
87 int main() {
88     string a, b, s;
89     int tt = 0;
90     while(getline(cin, a) && getline(cin, b)) {
91         if (tt++) cout << endl;
92         getline(cin, s);
93         s = a + "\1" + b + "\2";
94         suffix_array(s);
95         lcp(s);
96
97         int maxx = 0;
98         for(int i=1; i<s.size(); i++) {
99             bool left = SA[i-1]+LCP[i] <= a.size();
100             bool right = SA[i]+LCP[i] <= a.size();
101
102             if (LCP[i] && (left^right)) {
103                 maxx = max(maxx, LCP[i]);
104             }
105         }
106
107         if (maxx == 0) {
108             cout << "No common sequence." << endl;
109             continue;
110         }
111
112         string last = "some invalid string";
113         for(int i=1; i<s.size(); i++) {
114             bool left = SA[i-1]+LCP[i] <= a.size();
115             bool right = SA[i]+LCP[i] <= a.size();
116             string sub = s.substr(SA[i], maxx);
117
118             if (LCP[i]==maxx && (left^right) && last != sub) {
119                 cout << sub << endl;
120                 last = sub;
121             }

```



```

122 |     }
123 | }
124 | }

```

## uva/796.cpp

```

1  //796
2  //Critical Links
3  //Graphs;Finding Bridges
4  #include <iostream>
5  #include <cstring>
6  #include <string>
7  #include <sstream>
8  #include <vector>
9  #include <algorithm>
10 #define MAX 101
11 using namespace std;
12 int G[MAX][MAX], V[MAX], L[MAX], n, gpe;
13
14 struct Ponte {
15     int a, b;
16     Ponte() { }
17     Ponte(int a, int b) : a(min(a, b)), b(max(a, b)) {}
18 };
19 bool comp(const Ponte& a, const Ponte& b) { return a.a < b.a || (a.a==b.a && a.b < b.b); }
20 vector<Ponte> P;
21
22 void dfs(int u, int v) {
23     V[v] = L[v] = ++gpe;
24     for(int i = 0; i < n; i++) {
25         if(G[v][i]) {
26             if(!V[i]){
27                 dfs(v, i);
28                 L[v] = min(L[v], L[i]);
29                 if(L[i] > V[v]) P.push_back(Ponte(v, i));
30             } else if(i != u) {
31                 L[v] = min(L[v], V[i]);
32             }
33         }
34     }
35 }
36
37 int main() {
38     while(cin >> n) {
39         memset(G, 0, sizeof(G));
40         memset(V, 0, sizeof(V));
41         memset(L, 0, sizeof(L));
42         P.clear();
43         gpe = 0;
44
45         int a, an, b; char skip;
46         for(int i=0; i<n; i++) {
47             cin >> a >> skip >> an >> skip;
48             while(an--) {
49                 cin >> b; G[b][a] = G[a][b] = 1;
50             }
51         }
52
53         for(int i=0; i<n; i++)
54             if (!V[i])
55                 dfs(i, i);
56
57         cout << P.size() << " critical links" << endl;
58         sort(P.begin(), P.end(), comp);
59
60         for(int i=0; i<P.size(); i++) {
61             cout << P[i].a << " - " << P[i].b << endl;
62         }
63         cout << endl;
64     }
65 }

```

## uva/811.cpp

```

1  //811
2  //The Fortified Forest
3  //Math;Geometry;Convex Hull;Monotone Chain
4  #include <iostream>
5  #include <cmath>
6  #include <iomanip>
7  #include <algorithm>

```

```

8  #define EP 1e-6
9  using namespace std;
10
11 struct Tree {
12     int x, y, v, l;
13
14     int product(Tree a, Tree b) {
15         return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
16     }
17
18     bool left(Tree a, Tree b) {
19         return product(a, b) < 0;
20     }
21
22     double dist(Tree b) {
23         return sqrt(pow(x-b.x, 2.0) + pow(y-b.y, 2.0));
24     }
25
26     bool operator <(const Tree& p) const {
27         if (this->x != p.x) return this->x < p.x;
28         return this->y < p.y;
29     }
30
31     bool operator ==(const Tree& p) const {
32         return this->x == p.x and this->y == p.y;
33     }
34 };
35
36 int convexHull(Tree* P, int n, Tree* S) {
37     sort(P, P+n);
38
39     int m=0;
40     for(int i=0; i<n; i++) {
41         while(m >= 2 && S[m-1].left(S[m-2], P[i])) m--;
42         S[m++] = P[i];
43     }
44     m--;
45
46     for(int i=n-1, k=m; i >= 0; i--) {
47         while(m >= k+2 && S[m-1].left(S[m-2], P[i])) m--;
48         S[m++] = P[i];
49     }
50     m--;
51
52     return m;
53 }
54
55 Tree P[20], T[20], S[20];
56
57 int main() {
58     int n, tt=0;
59     while(cin >> n, n) {
60         for(int i=0; i<n; i++) {
61             cin >> P[i].x >> P[i].y >> P[i].v >> P[i].l;
62         }
63         int mini = 0, minv = 1<<30, minc = 1<<30;
64         double mine = 0.0;
65
66         for(int i=0; i<(1<<n); i++) {
67             int value = 0, length = 0, count=0, ts=0;
68             for(int j=0; j<n; j++) {
69                 if (i & 1<<j) {
70                     value += P[j].v;
71                     length += P[j].l;
72                     count++;
73                 } else {
74                     T[ts++] = P[j];
75                 }
76             }
77             if (value > minv) continue;
78             int s = convexHull(T, ts, S);
79
80             double perimeter = 0;
81             for(int j=0; j<s; j++)
82                 perimeter += S[j].dist(S[(j+1)%n]);
83
84             if (length > perimeter-EP && (value < minv || value == minv && count < minc)) {
85                 mini = i;
86                 minc = count;
87                 minv = value;
88                 mine = length - perimeter;
89             }
90         }
91     }

```

```

92         if (tt) cout << endl;
93         cout << "Forest " << ++tt << endl;
94         cout << "Cut these trees:";
95         for(int i=0; i<n; i++)
96             if (mini & 1<<i)
97                 cout << " " << i+1;
98         cout << endl;
99         cout << "Extra wood: " << fixed << setprecision(2) << mine << endl;
100     }
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             }
30         }
31     }
32
33     return 0;
34 }
35
36 int main() {
37     int tt=0;
38     while(cin >> n, n) {
39         memset(G, 0, sizeof(G));
40         memset(F, 0, sizeof(F));
41         memset(V, 0, sizeof(V));
42
43         tt++;
44         int s, t, c;
45         cin >> s >> t >> c;
46         for(int i=0; i<c; i++) {
47             int a, b, f;
48             cin >> a >> b >> f;
49             G[a][b] = G[b][a] += f;
50         }
51
52         int total = 0;
53         while(int sent = send(s, t, INT_MAX)) {
54             total += sent;
55             memset(V, 0, sizeof(V));
56         }
57
58         cout << "Network " << tt << endl;
59         cout << "The bandwidth is " << total << "." << endl;
60         cout << endl;
61     }
62 }
63
64
65 }

```

## uva/825.cpp

```
1 //825
2 //Walking on the Safe Side
3 //Dynamic Programming;Ad hoc
4 #include <iostream>
5 #include <string>
6 #include <sstream>
7 #include <cstring>
8 #define MAX 250
9 using namespace std;
10
11 int T[MAX][MAX];
12 int B[MAX][MAX];
13
14 int main() {
15     int t; cin >> t;
16     for(int tt=1; tt<=t; tt++) {
17         int r, c; cin >> r >> c;
18
19         string s;
20         getline(cin, s);
21         for(int i=0; i<r; i++) {
22             getline(cin, s);
23             stringstream sin(s);
24
25             int a, b; sin >> a;
26             while(sin >> b) {
27                 B[a-1][b-1] = tt;
28             }
29         }
30
31         for(int i=0; i<r; i++) {
32             for(int j=0; j<c; j++) {
33                 int s = (i==0 && j==0 ? 1 : 0);
34                 if (i>0) s+= T[i-1][j];
35                 if (j>0) s+= T[i][j-1];
36                 T[i][j] = (B[i][j] != tt ? s : 0);
37             }
38         }
39
40         if (tt>1) cout << endl;
41         cout << T[r-1][c-1] << endl;
42     }
43 }
```

## uva/902.cpp

```
1 //902
2 //Password Search
3 //Misc;STL map
4 #include <iostream>
5 #include <string>
6 #include <map>
7 using namespace std;
8
9 map<string, int> T;
10
11 int main() {
12     int n; string s;
13     while(cin >> n >> s) {
14         T.clear();
15
16         int maxx=0, maxv=0;
17         for(int i=0; i<=s.size()-n; i++){
18             int v = ++T[s.substr(i, n)];
19             if (v > maxv) {
20                 maxv = v;
21                 maxx = i;
22             }
23         }
24         cout << s.substr(maxx, n) << endl;
25     }
26 }
```

## uva/907.cpp

```
1 //907
2 //Winterim Backpacking Trip
```

```

3 //Dynamic Programming;Integer partition
4 #define MAX 602
5 #include <iostream>
6 #include <cstring>
7 #include <climits>
8 using namespace std;
9
10 int T[MAX][MAX], S[MAX], n, k;
11
12 int main() {
13     while(cin >> n >> k) {
14         n++; k++;
15         memset(T, 0, sizeof(T));
16
17         S[0] = 0;
18         for(int i=1; i<=n; i++) {
19             cin >> S[i];
20         }
21
22         for(int i=1; i<=n; i++)
23             T[i][1] = T[i-1][1]+S[i];
24
25         for(int i=1; i<=k; i++)
26             T[1][i] = S[1];
27
28         for(int i=2; i<=n; i++) {
29             for(int j=2; j<=k; j++) {
30                 T[i][j] = INT_MAX;
31                 for(int x=1; x<i; x++)
32                     T[i][j] = min(T[i][j], max(T[x][j-1], T[i][1] - T[x][1]));
33             }
34         }
35
36         cout << T[n][k] << endl;
37     }
38 }

```

## uva/908.cpp

```

1 //908
2 //Re-connecting Computer Sites
3 //Graphs;Minimum Spanning Tree;Prim
4 #include <iostream>
5 #include <cstring>
6 #include <climits>
7 #include <vector>
8 #include <algorithm>
9 #include <queue>
10 #define MAX 200010
11
12 struct Road {
13     int v, c;
14     Road(int v, int c) : v(v), c(c) {}
15     bool operator < (const Road& that) const { return c > that.c; }
16 };
17
18 using namespace std;
19
20 vector<Road> G[MAX];
21
22 int main() {
23     int n, k, m, t=0;
24     while(cin >> n) {
25         memset(G, 0, sizeof(G));
26
27         int before = 0;
28         for(int i=0; i<n-1; i++) {
29             int a, b, c;
30             cin >> a >> b >> c;
31             G[a].push_back(Road(b, c));
32             G[b].push_back(Road(a, c));
33             before += c;
34         }
35
36         int total=0;
37         cin >> k;
38         for(int i=0; i<k; i++) {
39             int a, b, c;
40             cin >> a >> b >> c;
41
42             int maxx = 0, maxv, side, counter;
43             for(int j=0; j<G[a].size(); j++)
44                 if (G[a][j].c > maxx) {

```

```

45         maxx = G[a][j].c;
46         maxv = j;
47         side= a; counter = b;
48     }
49
50     for(int j=0; j<G[b].size(); j++)
51     if (G[b][j].c > maxx) {
52         maxx = G[b][j].c;
53         maxv = j;
54         side= b; counter = a;
55     }
56
57     if (maxx <= c) continue;
58     total = maxx-c;
59     G[side][maxv].v = counter;
60     G[side][maxv].c = c;
61 }
62
63 cin >> m;
64 for(int i=0; i<m; i++) {
65     int a, b, c;
66     cin >> a >> b >> c;
67 }
68
69 if (t++) cout << endl;
70 cout << before << endl << before-total << endl;
71 }
72 return 0;
73 }

```

## uva/929.cpp

```

1 //929
2 //Number Maze
3 //Graphs;Shortest Path;Dijkstra
4 #include <iostream>
5 #include <cstring>
6 #include <queue>
7 #include <algorithm>
8 #define MAX 1001
9 using namespace std;
10
11 struct Edge {
12     int x, y, c;
13     Edge(int x, int y, int c) : x(x), y(y), c(c) {};
14     inline bool operator <(const Edge& a) const {
15         return c > a.c;
16     }
17 };
18
19 int G[MAX][MAX], V[MAX][MAX];
20 int n, m;
21 priority_queue<Edge> Q;
22
23 void try_q(int x, int y, int c) {
24     if (x <= 0 || x > n || y<=0 || y > m || c+G[x][y] >= V[x][y]) return;
25     Q.push(Edge(x, y, c+G[x][y]));
26 }
27
28
29 int main() {
30     int T; cin >> T;
31     for(int tt=1; tt<=T; tt++) {
32         memset(V, 0x7f, sizeof(V));
33         Q = priority_queue<Edge>();
34
35         cin >> n >> m;
36         for(int i=1; i<=n; i++)
37             for(int j=1; j<=m; j++)
38                 cin >> G[i][j];
39
40         Q.push(Edge(1, 1, G[1][1]));
41
42         while(!Q.empty()) {
43             Edge e = Q.top(); Q.pop();
44             if (V[e.x][e.y] <= e.c) continue;
45
46             V[e.x][e.y] = e.c;
47
48             if (e.x == n && e.y == m) break;
49
50             try_q(e.x-1, e.y, e.c);
51             try_q(e.x+1, e.y, e.c);

```

```

52         try_q(e.x, e.y-1, e.c);
53         try_q(e.x, e.y+1, e.c);
54     }
55
56     cout << V[n][m] << endl;
57 }
58 }

```

## uva/986.cpp

```

1  //986
2  //How Many?
3  //Dynamic Programming;Ad hoc
4  #include <iostream>
5  #include <cstring>
6  using namespace std;
7
8  int T[50][50][50][2];
9
10 int main() {
11     int n, r, h;
12
13     while(cin >> n >> r >> h) {
14         memset(T, 0, sizeof(T));
15         T[0][0][0][0] = 1;
16
17         for(int i=1; i<=2*n; i++) {
18             for(int j=0; j<=2*n; j++) {
19                 for(int k=0; k<=2*n; k++) {
20                     if (j<2*n) {
21                         T[i][j][k][0] += T[i-1][j+1][k][0];
22
23                         if (j+1==h && k>0)
24                             T[i][j][k][0] += T[i-1][j+1][k-1][1];
25                         else if (j+1!=h)
26                             T[i][j][k][0] += T[i-1][j+1][k][1];
27                     }
28
29                     if (j>0) {
30                         T[i][j][k][1] += T[i-1][j-1][k][0] + T[i-1][j-1][k][1];
31                     }
32                 }
33             }
34         }
35     }
36
37     cout << T[n*2][0][r][0] << endl;
38 }
39 }
40 }

```

## uva/1016.cpp

```

1  //1016
2  //Silly Sort
3  //Misc;Permutation Cycle
4  #include <iostream>
5  #include <algorithm>
6  #include <cstring>
7  #define MAX 2000
8  using namespace std;
9
10 int T[MAX], Q[MAX], M[MAX];
11 bool V[MAX];
12
13 int main() {
14     int n, t=0;
15     while(cin >> n, n) {
16         memset(V, 0, sizeof(V));
17
18         int minn=1<<30;
19         for(int i=0; i<n; i++) {
20             cin >> T[i];
21             minn = min(minn, T[i]);
22         }
23
24         memcpy(Q, T, sizeof(T));
25         sort(Q, Q+n);
26         for(int i=0; i<n; i++) {
27             M[Q[i]] = i;
28         }
29     }
30 }

```

```

29         int answer = 0;
30         for(int i=0; i<n; i++) {
31             if (V[i]) continue;
32
33             int cycle = 0, minc=1<<30, sumc=0;
34             for(int j=i; !V[j]; j=M[T[j]]) {
35                 V[j] = true;
36                 sumc += T[j];
37                 minc = min(minc, T[j]);
38                 cycle++;
39             }
40
41             answer += min(sumc + (cycle-2)*minc, sumc + minc + minn * (cycle+1));
42         }
43     }
44     cout << "Case " << ++t << ": " << answer << endl << endl;
45 }
46 }
47 }

```

## uva/1056.cpp

```

1  //1056
2  //Degrees of Separation
3  //Graphs;Shortest Path;Floyd-Warshall
4  #include <iostream>
5  #include <cstring>
6  #include <string>
7  #include <map>
8  #include <cassert>
9  #define MAX 51
10
11 using namespace std;
12
13 int G[MAX][MAX];
14 int n, m;
15 map<string, int> M;
16
17 int person(string& s) {
18     if (M.find(s) != M.end())
19         return M[s];
20     else
21         return M[s]=M.size();
22 }
23
24
25 int main() {
26     int t=0;
27     while(cin >> n >> m, n|m) {
28         memset(G, 0x1f, sizeof(G));
29         M.clear();
30
31         for(int i=0; i<m; i++) {
32             string p, q;
33             cin >> p >> q;
34             int a = person(p), b=person(q);
35             G[a][b] = G[b][a] = 1;
36         }
37         for(int i=1; i<=n; i++) G[i][i] = 0;
38
39         assert(M.size() <= n);
40
41         for(int k=1; k<=n; k++)
42             for(int i=1; i<=n; i++)
43                 for(int j=1; j<=n; j++)
44                     G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
45
46         int maxx = 0;
47         for(int i=1; i<=n; i++)
48             for(int j=1; j<=n; j++)
49                 maxx = max(maxx, G[i][j]);
50
51         cout << "Network " << ++t << ": ";
52
53         if (maxx <= n)
54             cout << maxx << endl;
55         else
56             cout << "DISCONNECTED" << endl;
57
58         cout << endl;
59
60     }
61 }

```



```

62 |     return 0;
63 | }

```

## uva/1158.cpp

```

1 | //1158
2 | //CubesSquared
3 | //Dynamic Programming;Knapsack;Infinite Items Knapsack
4 | #include <iostream>
5 | #include <cstring>
6 | #include <vector>
7 | using namespace std;
8 |
9 | int K[400001];
10 | vector<int> W;
11 |
12 | int main() {
13 |
14 |     for(int i=1; i*i*i<=400000; i++)
15 |         W.push_back(i*i*i);
16 |
17 |     for(int a=1, i=1; a<=400000; i++, a+=i*i)
18 |         W.push_back(a);
19 |
20 |     memset(K, 0x3f, sizeof(K));
21 |     K[0] = 0;
22 |     for(int i=0; i<W.size(); i++)
23 |         for(int j=W[i]; j<=400000; j++)
24 |             K[j] = min(K[j], K[j-W[i]]+1);
25 |
26 |     int n;
27 |     while(cin >> n, n!=-1)
28 |         cout << K[n] << endl;
29 |
30 |     return 0;
31 | }

```

## uva/1174.cpp

```

1 | //1174
2 | //IP-TV
3 | //Graphs;Minimum Spanning Tree;Prim;Priority Queue
4 | #include <iostream>
5 | #include <cstring>
6 | #include <climits>
7 | #include <string>
8 | #include <vector>
9 | #include <algorithm>
10 | #include <queue>
11 | #include <map>
12 | #define MAX 200010
13 |
14 | using namespace std;
15 |
16 | struct Road {
17 |     int v, c;
18 |     Road(int v, int c) : v(v), c(c) {}
19 |     inline bool operator < (const Road& that) const { return c > that.c; }
20 | };
21 |
22 | vector<Road> G[MAX];
23 | priority_queue<Road> Q;
24 | int n, m;
25 | bool V[MAX];
26 | map<string, int> M;
27 |
28 | int city(string& s) {
29 |     if (M.find(s) != M.end())
30 |         return M[s];
31 |     else
32 |         return M[s]=M.size();
33 | }
34 |
35 |
36 | int main() {
37 |     int t; cin >> t; t=0;
38 |
39 |     while(cin >> n >> m) {
40 |         memset(V, 0, sizeof(V));
41 |         memset(G, 0, sizeof(G));
42 |         M.clear();

```

```

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

```

## uva/1197.cpp

```

1  //1197
2  //The Suspects
3  //Graphs;DFS
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <vector>
8  using namespace std;
9
10 vector<int> G[501], P[30001];
11 bool VG[501], VP[30001];
12 int n, m;
13
14 int dfs(int v) {
15     int sum = 1;
16     VP[v] = true;
17
18     for(int i=0; i<P[v].size(); i++) {
19         int g = P[v][i];
20         if (VG[g]) continue;
21         VG[g] = true;
22
23         for(int j=0; j<G[g].size(); j++) {
24             int u = G[g][j];
25             if (VP[u]) continue;
26             sum += dfs(u);
27         }
28     }
29
30     return sum;
31 }
32
33 int main() {
34     while(cin >> n >> m, n|m) {
35         memset(G, 0, sizeof(G));
36         memset(P, 0, sizeof(P));
37         memset(VG, 0, sizeof(VG));
38         memset(VP, 0, sizeof(VP));
39
40         for(int i=0; i<m; i++) {
41             int k; cin >> k;
42             while(k--) {
43                 int a; cin >> a;
44                 G[i].push_back(a);
45                 P[a].push_back(i);
46             }
47         }
48     }

```

```

49         cout << dfs(0) << endl;
50     }
51     return 0;
52 }

```

## uva/1200.cpp

```

1  //1200
2  //A DP Problem
3  //Misc;String parsing
4  #include <iostream>
5  #include <string>
6  #include <cmath>
7  using namespace std;
8
9  int getSign(string& s, int &i) {
10     if (s[i] == '+') { i++; return 1; }
11     if (s[i] == '-') { i++; return -1; }
12     return 1;
13 }
14
15 int getNumber(string& s, int& i, bool& got) {
16     int result = 0;
17     while(s[i] >='0' && s[i] <= '9') {
18         result = result*10 + (s[i]-'0');
19         i++;
20         got = true;
21     }
22     return result;
23 }
24
25 bool getX(string& s, int& i) {
26     return i<s.size() && s[i] == 'x' && ++i;
27 }
28
29 bool willChange(string& s, int& i) {
30     return i<s.size() && s[i] == '=' && ++i;
31 }
32
33
34 int main()
35 {
36     int t;
37     string s;
38     cin >> t;
39
40     while(cin >> s) {
41         int i=0, A=0, B=0, masterSign = 1;
42         while(i<s.size()) {
43             int sign = getSign(s, i);
44             bool got = false;
45             int number = getNumber(s, i, got);
46             bool isX = getX(s, i);
47             if (isX && !got) number = 1;
48             // cout << masterSign << " " << sign << " " << number << " " << isX << endl;
49
50             if (isX)
51                 B += -1*masterSign*sign*number;
52             else
53                 A += masterSign*sign*number;
54             if (willChange(s, i)) masterSign *= -1;
55         }
56         if (A==0 && B==0) {
57             cout << "IDENTITY" << endl;
58         } else if (B==0) {
59             cout << "IMPOSSIBLE" << endl;
60         } else {
61             cout << (int)floor(((double)A/B))<< endl;
62         }
63     }
64 }
65
66 return 0;
67 }

```

## uva/1203.cpp

```

1  //1203
2  //Argus
3  //Misc;Priority queue
4  #include<cstdio>

```

```

5  #include<iostream>
6  #include<queue>
7  #include<string>
8  using namespace std;
9
10 #define SZ 3200
11
12 struct Item{
13     int p, q, b;
14
15     Item() {}
16     Item(int q, int p) : p(p), q(q), b(p) {}
17     Item(int q, int p, int b) : p(p), q(q), b(b) {}
18
19     inline bool operator < (const Item &d) const{
20         if(this->p==d.p) return d.q<this->q;
21         return this->p>d.p;
22     }
23
24     Item next() {
25         return Item(q, p+b, b);
26     }
27 };
28
29 priority_queue<Item> Q;
30 int q, p;
31
32 int main(void) {
33     string s;
34     int q, p, k;
35
36     while(cin >> s, s!="#") {
37         cin >> q >> p;
38         Q.push(Item(q, p));
39     }
40
41     cin >> k;
42     for(int i=0; i<k; ++i) {
43         Item item = Q.top(); Q.pop();
44         cout << item.q << endl;
45         Q.push(item.next());
46     }
47
48     return 0;
49 }
50

```

## uva/1205.cpp

```

1  //1205
2  //Color a Tree
3  //Graphs;Job Scheduling
4  #include <iostream>
5  #include <vector>
6  #include <set>
7  #define MAX 1008
8  using namespace std;
9
10 struct Cost {
11     int a, b, t, v;
12     Cost() { }
13     Cost(int a, int b, int t, int v) : a(a), b(b), t(t), v(v) {}
14
15     inline bool operator <(const Cost &c) const {
16         int cra = a*c.t, crb = c.a*t;
17         if (cra!=crb) return cra>crb;
18         return v<c.v;
19     }
20 };
21
22 int P[MAX], M[MAX];
23 set<Cost> S;
24 Cost C[MAX];
25
26 int findParent(int v) {
27     if (M[v] == v) return v;
28     return M[v] = findParent(M[v]);
29 }
30
31 int main() {
32     int n, r;
33     while(cin >> n >> r, n|r) {
34         for(int i=1; i<=n; i++) {

```

```

35         int a; cin >> a;
36
37         P[i] = 0;
38         M[i] = i;
39         C[i] = *S.insert(Cost(a, a, 1, i)).first;
40     }
41
42     for(int i=1; i<=n-1; i++) {
43         int u, v; cin >> u >> v;
44         P[v] = u;
45     }
46
47     int total = 0, time = 0;
48     while(!S.empty()) {
49         Cost c = *S.begin();
50         int pid = findParent(P[c.v]);
51         if (pid == 0) {
52             total += time * c.a + c.b;
53             time += c.t;
54             S.erase(c);
55             M[c.v] = 0;
56         } else {
57             Cost d = *S.find(C[pid]);
58             Cost e(c.a + d.a, c.b + d.b + c.a * d.t, c.t + d.t, d.v);
59
60             S.erase(c);
61             S.erase(d);
62             S.insert(e);
63
64             M[c.v] = d.v;
65             C[e.v] = e;
66         }
67     }
68     cout << total << endl;
69
70 }
71 }

```

## uva/1207.cpp

```

1  //1207
2  //AGTC
3  //Dynamic Programming;Edit Distance
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #include <cmath>
8  #define MAX 1005
9  using namespace std;
10
11  int T[MAX][MAX];
12  string P, Q;
13
14
15  int main() {
16      int p, q;
17      while(cin >> p >> P >> q >> Q) {
18          for(int i=0; i<=p; i++) { T[i][0] = i; }
19          for(int i=0; i<=q; i++) { T[0][i] = i; }
20
21          for(int i=1; i<=p; i++) {
22              for(int j=1; j<=q; j++) {
23                  if (P[i-1] == Q[j-1])
24                      T[i][j] = T[i-1][j-1];
25                  else
26                      T[i][j] = min(min(T[i-1][j], T[i][j-1]), T[i-1][j-1])+1;
27              }
28          }
29
30          cout << T[p][q] << endl;
31      }
32
33      return 0;
34  }

```

## uva/1208.cpp

```

1  //1208
2  //Oreon
3  //Graphs;Minimum Spanning Tree;Prim
4  #include <iostream>

```

```

5  #include <cstring>
6  #include <climits>
7  #include <vector>
8  #include <algorithm>
9  #define MAX 501
10
11 using namespace std;
12
13 int G[MAX][MAX], n;
14 bool V[MAX];
15 int D[MAX], DO[MAX];
16
17 struct Item {
18     int p, a, b;
19     Item(){}
20     Item(int p, int a, int b) : p(p), a(min(a,b)), b(max(a,b)) {}
21 };
22
23 bool comp(const Item& a, const Item& b) {
24     if (a.p != b.p) return a.p < b.p;
25     if (a.a != b.a) return a.a < b.a;
26     if (a.b != b.b) return a.b < b.b;
27     return false;
28 }
29
30 vector<Item> R;
31
32 int updateD(int i) {
33     D[i] = 0;
34     for(int j=0; j<n; j++) {
35         if (G[i][j] && G[i][j] < D[j]) {
36             D[j] = G[i][j];
37             DO[j] = i;
38         }
39     }
40 }
41
42 int main() {
43     int t; char skip;
44     cin >> t;
45     t = 0;
46     while(cin >> n) {
47         memset(V, 0, sizeof(V));
48         memset(D, 0x3F, sizeof(D));
49         R.clear();
50
51         for(int i=0; i<n; i++) {
52             for(int j=0; j<n; j++) {
53                 cin >> G[i][j];
54                 if (j+1<n) cin >> skip;
55             }
56         }
57
58         int total = 0;
59
60         V[0] = true;
61         updateD(0);
62
63         for(int k=1; k<n; k++) {
64             int minn=INT_MAX, minv;
65             for(int i=0; i<n; i++) {
66                 if (!V[i] && D[i] < minn) {
67                     minn = D[i];
68                     minv = i;
69                 }
70             }
71             R.push_back(Item(minn, DO[minv], minv));
72             V[minv] = true;
73             updateD(minv);
74             total += minn;
75         }
76
77         sort(R.begin(), R.end(), comp);
78         cout << "Case " << ++t << ":" << endl;
79         for(int i=0; i<R.size(); i++) {
80             cout << (char)(R[i].a+'A') << "-" << (char)(R[i].b+'A') << " " << R[i].p << endl;
81         }
82     }
83     return 0;
84 }

```

```

1 //1213
2 //Sum of Different Primes
3 //Dynamic Programming;Knapsack;Counting Knapsack
4 #include <iostream>
5 #include <vector>
6 #include <cstring>
7 using namespace std;
8
9 long K[20][1300];
10 bool P[1300];
11 vector<int> W();
12
13 int main() {
14     int n, k;
15
16     memset(P, true, sizeof(P));
17     P[0] = P[1] = false;
18     for(int i=2; i<1300; i++) {
19         if (P[i]) {
20             W.push_back(i);
21             for(int j=i*i; j<1300; j+=i)
22                 P[j] = false;
23         }
24     }
25
26     K[0][0] = 1;
27     for(int i=0; i<W.size(); i++)
28         for(int p=19; p>0; p--)
29             for(int j = W[i]; j<1300; j++)
30                 K[p][j]+=K[p-1][j-W[i]];
31
32     while(cin >> n >> k, n|k)
33         cout << K[k][n] << endl;
34 }

```

## uva/1215.cpp

```

1 //1215
2 //String Cutting
3 //Misc;Binary Search
4 #include <iostream>
5 #include <string>
6 #include <set>
7 using namespace std;
8
9 int T[10001][26];
10 int C[1001];
11 set<int> K;
12
13 int main() {
14     int t; cin >> t; t=0;
15     int k; string s;
16     while(cin >> k) {
17         K.clear();
18         for(int i=0; i<k; i++)
19             cin >> C[i];
20
21         cin >> s;
22         for(int i=1; i<=s.size(); i++)
23             for(int j=0; j<26; j++)
24                 T[i][j] = T[i-1][j] + (s[i-1] == j+'a');
25
26         K.insert(0);
27         K.insert(s.size());
28
29         int total = 0;
30         for(int i=0; i<k; i++) {
31             int mid = C[i];
32             set<int>::iterator it = K.lower_bound(mid);
33             int hi = *it; it--;
34             int lo = *it;
35
36             for(int j=0; j<26; j++) {
37                 int sidea = T[mid][j]-T[lo][j];
38                 int sideb = T[hi][j]-T[mid][j];
39
40                 if (sidea>0 ^ sideb>0) total++;
41             }
42             K.insert(mid);
43         }
44         cout << total << endl;
45     }
46 }

```

47 | }

## uva/1216.cpp

```
1 //1216
2 //The Bug Sensor Problem
3 //Graphs;DFS
4 #include <iostream>
5 #include <cstring>
6 #include <cmath>
7 #define MAX 1000
8 using namespace std;
9
10 double G[MAX][MAX];
11 int X[MAX], Y[MAX], n, k;
12 int V[MAX];
13
14 void dfs(int v, int comp, int max) {
15     V[v] = comp;
16     for(int i=0; i<n; i++) {
17         if (!V[i] && G[v][i] <= max)
18             dfs(i, comp, max);
19     }
20 }
21
22 int main() {
23     int t; cin >> t; t=0;
24     while(cin >> k) {
25         n=0;
26
27         double maxd=0;
28         while(cin >> X[n], X[n]!=-1) {
29             cin >> Y[n];
30             for(int i=0; i<n; i++) {
31                 G[i][n] = G[n][i] = sqrt(pow(X[n]-X[i], 2.0)+pow(Y[n]-Y[i], 2.0));
32                 maxd = max(maxd, G[i][n]);
33             }
34             n++;
35         }
36
37         int begin=0, end=(int)ceil(maxd);
38         int best, last = -1;
39         while(begin <= end) {
40             int mid = (begin+end)/2;
41             if (mid == last) break;
42
43             int comp=0;
44             memset(V, 0, sizeof(V));
45             for(int i=0; i<n; i++)
46                 if (!V[i])
47                     dfs(i, ++comp, mid);
48
49             last = mid;
50             if (comp > k)
51                 begin = mid;
52             else {
53                 if (comp == k) best = mid;
54                 end = mid;
55             }
56         }
57
58         cout << best << endl;
59     }
60 }
```

## uva/1220.cpp

```
1 //1220
2 //Party at Hali-Bula
3 //Graphs;DFS
4 #include <iostream>
5 #include <string>
6 #include <map>
7 #include <cstring>
8 #include <vector>
9 #define MAX 205
10 using namespace std;
11
12 map<string, int> E;
13 int emp(string& s) {
14     if (E.find(s) != E.end())
```



```

15         return E[s];
16     else
17         return E[s] = E.size()-1;
18 }
19
20 bool L[MAX], L2[MAX];
21 vector<int> G[MAX];
22 int n;
23
24 int dfs(int v) {
25     int acum = 0, illu = 0;
26     for(int i=0; i<G[v].size(); i++) {
27         acum += dfs(G[v][i]);
28         if (L[G[v][i]]) illu++;
29     }
30     if (G[v].size() > 0 && illu < G[v].size())
31         L[v] = true;
32     return acum + L[v];
33 }
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 (DFSst(O[i], comp) > 1 || GR[O[i]] == 0) {
90                 for(int j=1;j<=n;j++) {
91                     if (CO[j] == comp) {
92                         words.insert(j);
93                     }
94                 }
95             }
96         }
97     }
98
99     memset(V, 0, sizeof(V));
100    for(set<int>::iterator it=words.begin(); it!=words.end(); it++) {
101        DFSf(*it);
102    }
103
104    cout << answer.size() << endl;
105    for(set<string>::iterator it=answer.begin(); it!=answer.end(); it++)
106        cout << (it!=answer.begin()?" ":"") << *it;
107    cout << endl;
108 }
109
110 return 0;
111 }

```

## uva/1231.cpp

```

1  //1231
2  //ACORN
3  //Dynamic Programming;Ad hoc
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #define MAX 2001
8  using namespace std;
9
10 int S[MAX][MAX], T[MAX][MAX], M[MAX];
11 char skip;
12 int main() {
13     int cases; cin >> cases;
14     while(cases--) {
15         memset(S, 0, sizeof(S));
16         memset(M, 0, sizeof(M));
17         int t, h, f;
18         cin >> t >> h >> f;
19
20         for(int i=0; i<t; i++) {
21             int k, a; cin >> k;
22             while(k--) {
23                 cin >> a;
24                 S[a][i]++;
25             }
26         }
27
28         for(int i=h; i>=0; i--) {
29             for(int j=0; j<t; j++) {
30                 int move = i+f<=h ? M[i+f] : 0;
31                 int stay = i+1<=h ? T[i+1][j] : 0;
32                 T[i][j] = max(move, stay) + S[i][j];
33                 M[i] = max(M[i], T[i][j]);
34             }
35         }
36
37         cout << M[0] << endl;
38     }
39
40     return 0;
41 }

```

## uva/1232.cpp

```

1 //1232
2 //SKYLINE
3 //Misc;Segment Tree
4 #include <iostream>
5 #include <string>
6 #include <set>
7 using namespace std;
8
9 struct Node {
10     int a, b, h;
11     bool leaf;
12     Node() {}
13     Node(int a, int b, int h, bool leaf=true) : a(a), b(b), h(h), leaf(leaf) {}
14 };
15
16 Node H[500005];
17 inline int left(int i) { return 2*i; }
18 inline int right(int i) { return 2*i+1; }
19
20 inline void cut(int v, int x) {
21     H[left(v)] = Node(H[v].a, x, H[v].h);
22     H[right(v)] = Node(x, H[v].b, H[v].h);
23     H[v].leaf = false;
24 }
25
26 int dfs(int v, int a, int b, int h) {
27     a = max(a, H[v].a);
28     b = min(b, H[v].b);
29     if (b<=a) return 0;
30
31     if (!H[v].leaf)
32         return dfs(left(v), a, b, h) + dfs(right(v), a, b, h);
33
34     if (H[v].h > h) return 0;
35     if (H[v].a < a) return cut(v, a), dfs(v, a, b, h);
36     if (b < H[v].b) return cut(v, b), dfs(v, a, b, h);
37
38     H[v].h = h;
39     return b-a;
40 }
41
42 int main() {
43     int n, t; cin >> t; t=0;
44     while(cin >> n, n) {
45         H[1] = Node(0, 100000, 0);
46
47         int sum = 0;
48         while(n--) {
49             int a, b, h;
50             cin >> a >> b >> h;
51             sum += dfs(1, a, b, h);
52         }
53         cout << sum << endl;
54     }
55 }
56

```

## uva/1233.cpp

```

1 //1233
2 //USHER
3 //Graphs;Shortest Path;Floyd-Warshall
4 #include <iostream>
5 #include <cstring>
6 #include <string>
7 #include <map>
8 #include <cassert>
9 #define MAX 501
10
11 using namespace std;
12
13 int P[MAX];
14 int G[MAX][MAX];
15 int n, m;
16
17 int main() {
18     int tt; cin >> tt;
19     while(tt--) {
20         int b, p, q; cin >> b >> p >> q;
21         for(int i=0; i<q; i++)
22             cin >> P[i];
23
24         memset(G, 0x1f, sizeof(G));
25
26     }
27 }
28

```

```

25
26     G[0][0] = 0;
27     for(int i=1; i<=p; i++) {
28         int k; cin >> k;
29         G[i][i] = 0;
30         for(int j=0; j<k; j++) {
31             int x, y; cin >> x >> y;
32             G[i][y] = min(G[i][y], x);
33         }
34     }
35
36     for(int k=0; k<=p; k++)
37         for(int i=0; i<=p; i++)
38             for(int j=0; j<=p; j++)
39                 G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
40
41     int minn = 1<<30;
42     for(int i=0; i<q; i++) {
43         minn = min(minn, G[P[i]][0]);
44     }
45
46     int current = 0;
47     int answer = 0;
48     while(true) {
49         if ((current += minn) >= b) break;
50         current--;
51         answer++;
52     }
53
54     cout << answer << endl;
55 }
56 return 0;
57 }

```

## uva/1234.cpp

```

1 //1234
2 //RACING
3 //Graphs;Minimum Spanning Tree;Prim;Priority Queue
4 #include <iostream>
5 #include <cstring>
6 #include <climits>
7 #include <vector>
8 #include <algorithm>
9 #include <queue>
10 #define MAX 10005
11
12 using namespace std;
13
14 struct Road {
15     int v, c;
16     Road(int v, int c) : v(v), c(c) {}
17     inline bool operator < (const Road& that) const { return c < that.c; }
18 };
19
20 vector<Road> G[MAX];
21 int CStart[MAX], CCount[MAX], nc;
22 priority_queue<Road> Q;
23 vector<int> R;
24 int n, m;
25 bool V[MAX];
26
27 int dfs(int v) {
28     V[v] = true;
29     int acum = 1;
30     for(int i=0; i<G[v].size(); i++)
31         if (!V[G[v][i].v])
32             acum += dfs(G[v][i].v);
33     return acum;
34 }
35
36 int main() {
37     int t; cin >> t;
38     while(cin >> n >> m, t--) {
39         memset(V, 0, sizeof(V));
40         memset(G, 0, sizeof(G));
41         nc = 0;
42         R.clear();
43
44         for(int i=0; i<m; i++) {
45             int a, b, c;
46             cin >> a >> b >> c;
47             G[a].push_back(Road(b, c));

```

```

48         G[b].push_back(Road(a, c));
49     }
50
51     for(int i=1; i<=n; i++) {
52         if (!V[i]) {
53             CStart[nc]=i;
54             CCount[nc]=dfs(i);
55             nc++;
56         }
57     }
58
59     int result=0;
60     for(int i=0; i<nc; i++) {
61         int totalc=0;
62         Q.push(Road(CStart[i], 0));
63         memset(V, 0, sizeof(V));
64
65         while(totalc < CCount[i]) {
66             Road item = Q.top(); Q.pop();
67             if (V[item.v]) { result+=item.c; continue; }
68
69             V[item.v] = true;
70             totalc++;
71
72             for(int j=0; j<G[item.v].size(); j++)
73                 if (!V[G[item.v][j].v])
74                     Q.push(G[item.v][j]);
75         }
76         while(!Q.empty()) {
77             result += Q.top().c;
78             Q.pop();
79         }
80     }
81     cout << result << endl;
82 }
83 return 0;
84 }

```

## uva/1235.cpp

```

1 //1235
2 //Anti Brute Force Lock
3 //Graphs;Minimum Spanning Tree;Prim
4 #include <iostream>
5 #include <cstring>
6 #include <climits>
7 #define MAX 501
8
9 using namespace std;
10
11 int abs(int a) {
12     return a>0?a:-a;
13 }
14
15 int d(int a, int b) {
16     int result = 0;
17     for(int i=0; i<4; i++) {
18         int aa=a%10, bb=b%10;
19         result += min(abs(aa-bb), 10-abs(aa-bb));
20         a/=10; b/=10;
21     }
22     return result;
23 }
24
25 int K[MAX], G[MAX][MAX], n;
26 bool V[MAX];
27 int D[MAX];
28
29 int updateD(int i) {
30     D[i] = 0;
31     for(int j=0; j<n; j++) {
32         if (G[i][j]) D[j] = min(D[j], G[i][j]);
33     }
34 }
35
36 int main()
37 {
38     int t;
39     cin >> t;
40     while(cin >> n) {
41         memset(V, 0, sizeof(V));
42         memset(D, 0x3F, sizeof(D));
43     }

```

```

44     for(int i=0; i<n; i++)
45         cin >> K[i];
46
47     for(int i=0; i<n; i++)
48         for(int j=i+1; j<n; j++)
49             G[i][j] = G[j][i] = d(K[i],K[j]);
50
51     int total=INT_MAX;
52     for(int i=0;i<n;i++) total = min(total, d(0, K[i]));
53
54     V[0] = true;
55     updateD(0);
56
57     for(int k=1; k<n; k++) {
58         int minn=INT_MAX, minv;
59         for(int i=0; i<n; i++) {
60             if (!V[i] && D[i] < minn) {
61                 minn = D[i];
62                 minv = i;
63             }
64         }
65         V[minv] = true;
66         updateD(minv);
67         total += minn;
68     }
69
70     cout << total << endl;
71 }
72 return 0;
73 }

```

## uva/1239.cpp

```

1  //1239
2  //Greatest K-Palindrome Substring
3  //Dynamic Programming;Ad hoc
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #include <cmath>
8  #define MAX 1005
9  using namespace std;
10
11 int T[MAX][MAX];
12
13 int main() {
14     int t; cin >> t; t=t-1;
15     string P;
16     int k;
17     while(cin >> P >> k) {
18         int p = P.size();
19
20         int maxx=0;
21         for(int i=p; i>=1; i--) {
22             for(int j=i; j<=p; j++) {
23                 T[i][j] = T[i+1][j-1] + (P[i-1] == P[j-1] ? 0 : 1);
24
25                 if (T[i][j] <= k)
26                     maxx = max(maxx, j-i+1);
27             }
28         }
29
30         cout << maxx << endl;
31     }
32
33     return 0;
34 }

```

## uva/1246.cpp

```

1  //1246
2  //Find Terrorists
3  //Math;Sieve
4  #include <iostream>
5  #include <vector>
6  #include <cstring>
7  using namespace std;
8
9  bool P[100];
10 int T[10000001];
11 vector<int> W;

```

```

12
13 long long real_mod(long long a, long long b) {
14     long long c = a%b;
15     if (c<0) c+=b;
16     return c;
17 }
18
19 int main() {
20     int n, k;
21
22     memset(P, true, sizeof(P));
23     P[0] = P[1] = false;
24     for(int i=2; i<100; i++) {
25         if (P[i]) {
26             W.push_back(i);
27             for(int j=i*i; j>=0 && j<100; j+=i)
28                 P[j] = false;
29         }
30     }
31
32     int t; cin >> t; t=0;
33     int a, b;
34     while(cin >> a >> b) {
35         memset(T, 0, sizeof(int)*(b-a+1));
36
37         if (a==0) { T[0]-=2; T[1] -= 1; }
38         if (a==1) { T[0]-=1; }
39
40         for(long long i=2; i*i<=b; i++) {
41             for(long long j=max(real_mod(i*i+i-a, i), i*i+i-a); j<=(b-a); j+=i) {
42                 T[j]+=2;
43             }
44             int tmp = i*i-a;
45             if (tmp >= 0 && tmp <= (b-a))
46                 T[tmp]++;
47         }
48
49         int cnt=0;
50         for(int i=0; i<=(b-a);i++) {
51             if (P[T[i]+2]) {
52                 cout << (cnt++?" ":"") << i+a;
53             }
54         }
55         if (!cnt) cout << -1;
56         cout << endl;
57     }
58
59 }

```

## uva/1247.cpp

```

1 //1247
2 //Interstar Transport
3 //Graphs;Shortest Path;Dijkstra
4 #include <iostream>
5 #include <cstring>
6 #include <climits>
7 #include <vector>
8 #include <algorithm>
9 #include <queue>
10 #define MAX 30
11
12 using namespace std;
13
14 struct Edge {
15     int u, v, c;
16     Edge(int u, int v, int c) : u(u), v(v), c(c) {}
17     inline bool operator < (const Edge& that) const { return c > that.c; }
18 };
19
20 int G[MAX][MAX];
21 int V[MAX];
22 int D[MAX];
23 int n, m;
24
25 void show(int t) {
26     if (D[t] != t) {
27         show(D[t]);
28         cout << " ";
29     }
30
31     cout << (char)(t+'A');
32 }

```



```

33
34 int shortest(int a, int b) {
35     memset(V, 0x3f, sizeof(V));
36     priority_queue<Edge> Q;
37     Q.push(Edge(a, a, 0));
38
39     while(!Q.empty()) {
40         Edge item = Q.top(); Q.pop();
41         if (item.c >= V[item.v]) continue;
42         V[item.v] = item.c;
43         D[item.v] = item.u;
44
45         for(int j=0; j<n; j++) {
46             if (G[item.v][j]) {
47                 Edge e = Edge(item.v, j, item.c+G[item.v][j]);
48                 if (e.c <= V[e.v])
49                     Q.push(e);
50             }
51         }
52     }
53     show(b); cout << endl;
54 }
55
56 int main() {
57     while(cin >> n >> m) {
58         memset(G, 0, sizeof(G));
59
60         for(int i=0; i<m; i++) {
61             char a, b; int c;
62             cin >> a >> b >> c;
63             G[a-'A'][b-'A'] = G[b-'A'][a-'A'] = c;
64         }
65
66         int k; cin >> k;
67         while(k--) {
68             char a, b; cin >> a >> b;
69             shortest(a-'A', b-'A');
70         }
71     }
72     return 0;
73 }

```

## uva/1251.cpp

```

1  //1251
2  //Repeated Substitution with Sed
3  //Graphs;Shortest Path;BFS
4  #include <iostream>
5  #include <queue>
6  #include <string>
7  #include <set>
8  #define MAX 1000
9  using namespace std;
10
11 struct Item {
12     string s;
13     int c;
14     Item(string s, int c) : s(s), c(c) {}
15 };
16
17 string replace(string str, string from, string to) {
18     if(from.empty())
19         return str;
20     int start_pos = 0;
21     while((start_pos = str.find(from, start_pos)) != string::npos) {
22         str.replace(start_pos, from.length(), to);
23         start_pos += to.length();
24     }
25     return str;
26 }
27
28 int n;
29 string A[MAX], B[MAX];
30
31 int main() {
32     while(cin >> n, n) {
33         for(int i=0; i<n; i++)
34             cin >> A[i] >> B[i];
35         string a, b;
36         cin >> a >> b;
37
38         queue<Item> Q;
39         set<string> S;

```

```

40     Q.push(Item(a, 0));
41
42     int answer = -1;
43     while(!Q.empty()) {
44         Item e = Q.front(); Q.pop();
45         if (e.s == b) {
46             answer = e.c;
47             break;
48         }
49
50         if (S.find(e.s) != S.end()) continue;
51         S.insert(e.s);
52
53         for(int i=0; i<n; i++) {
54             string s = replace(e.s, A[i], B[i]);
55             if (S.find(s) != S.end() || s.size() > 10) continue;
56             Q.push(Item(s, e.c+1));
57         }
58     }
59
60     cout << answer << endl;
61 }
62
63 }
64

```

## uva/1263.cpp

```

1  //1263
2  //Mines
3  //Graphs;Topological Sorting
4  #include <iostream>
5  #include <cstring>
6  #define MAX 2001
7  using namespace std;
8
9  int n;
10 bool V[MAX], G[MAX][MAX];
11 int X[MAX], Y[MAX], D[MAX], O[MAX], npv;
12
13 inline int abs(int a) {
14     return a>0?a:-a;
15 }
16
17 void dfs(int v, bool sort){
18     V[v] = true;
19     for(int i = 1; i <= n; i++)
20         if (G[v][i] && !V[i])
21             dfs(i, sort);
22     if (sort)
23         O[++npv] = v;
24 }
25
26 int main() {
27     int t; cin >> t; t=0;
28     while(cin >> n) {
29         memset(G, 0, sizeof(G));
30         for(int i=1; i<=n; i++) {
31             cin >> X[i] >> Y[i] >> D[i];
32         }
33         for(int i=1; i<=n; i++) {
34             for(int j=1; j<=n; j++) {
35                 int r = D[i]/2;
36                 if (abs(X[j]-X[i])<=r && abs(Y[j]-Y[i]) <=r && i!=j)
37                     G[i][j] = true;
38             }
39         }
40
41         npv = 0;
42         memset(V, 0, sizeof(V));
43         memset(O, 0, sizeof(O));
44
45         for(int i = 1; i <= n; i++)
46             if(!V[i]) dfs(i, true);
47
48         memset(V, 0, sizeof(V));
49
50         int comp = 0;
51         for(int i = n; i > 0; i--)
52             if(!V[O[i]]) {
53                 comp++;
54                 dfs(O[i], false);
55             }
56     }
57 }

```

```

56 |
57 |         cout << comp << endl;
58 |     }
59 |
60 |     return 0;
61 | }

```

## uva/1265.cpp

```

1 | //1265
2 | //Tour Belt
3 | //Graphs;Minimum Spanning Tree;Kruskal
4 | #include <iostream>
5 | #include <cstring>
6 | #include <vector>
7 | #include <set>
8 | #include <algorithm>
9 | #include <cassert>
10 | using namespace std;
11 |
12 | struct Edge {
13 |     int x, y, v;
14 |     Edge() {}
15 |     Edge(int x, int y, int v) : x(x), y(y), v(v) {}
16 |
17 |     inline bool operator <(const Edge& that) const {
18 |         return this->v > that.v;
19 |     }
20 | };
21 |
22 | Edge E[5006*2506];
23 | int A[5006][5006], B[5006][5006];
24 | int P[5002], C[5002];
25 |
26 | inline int findset(int v) {
27 |     if (P[v] != v)
28 |         return P[v] = findset(P[v]);
29 |     return v;
30 | }
31 |
32 | inline int unionset(int x, int y) {
33 |     int a = findset(x), b = findset(y);
34 |     if (a==b) return 0;
35 |     if (a>b) swap(a,b);
36 |     P[b] = a;
37 |     C[a] += C[b];
38 |     C[b] = 0;
39 |     return a;
40 | }
41 |
42 | int main() {
43 |     int tt; cin >> tt;
44 |
45 |     while(tt--) {
46 |         int n, m; cin >> n >> m;
47 |         for(int i=1; i<=n; i++) {
48 |             P[i] = i;
49 |             C[i] = 1;
50 |
51 |             for(int j=1; j<=n; j++) {
52 |                 A[i][j] = 1<<29;
53 |                 B[i][j] = 0;
54 |             }
55 |         }
56 |
57 |         for(int i=0; i<m; i++) {
58 |             int x, y, v;
59 |             cin >> x >> y >> v;
60 |             E[i] = Edge(x, y, v);
61 |
62 |             A[x][y] = A[y][x] = B[x][y] = B[y][x] = v;
63 |         }
64 |
65 |         sort(E, E+m);
66 |
67 |         int total = 0;
68 |         for(int i=0; i<m; i++) {
69 |             int x = findset(E[i].x), y = findset(E[i].y);
70 |             if (x==y) continue;
71 |
72 |             int a = unionset(x, y);
73 |
74 |             int outside = 0, inside = 1<<29;

```

```

75         for(int j=1; j<=n; j++) {
76             A[a][j] = A[j][a] = min(A[x][j], A[y][j]);
77             B[a][j] = B[j][a] = max(B[x][j], B[y][j]);
78
79             if (findset(a) == findset(j))
80                 inside = min(inside, A[a][j]);
81             else
82                 outside = max(outside, B[a][j]);
83         }
84
85         if (inside > outside)
86             total += C[a];
87     }
88
89     cout << total << endl;
90
91 }
92 }

```

## uva/10003.cpp

```

1  //10003
2  //Cutting Sticks
3  //Dynamic Programming;Matrix Multiplication
4  #define MAX 1001
5  #include <iostream>
6  #include <cstring>
7  #include <limits>
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 (a+1==b) return 0;
15     if (V[a][b]) return T[a][b];
16
17     int minn = INT_MAX;
18     for(int i=a+1; i<b; i++)
19         minn = min(minn, TT(a,i) + TT(i,b) + S[b]-S[a]);
20
21     V[a][b] = true;
22     return T[a][b] = minn;
23 }
24
25 int main() {
26     int t;
27     while(cin >> t, t) {
28         cin >> n;
29
30         memset(S, 0, sizeof(S));
31         memset(V, 0, sizeof(V));
32         S[0] = 0;
33         for(int i=1; i<=n; i++) {
34             cin >> S[i];
35         }
36         S[n+1] = t;
37
38         cout << "The minimum cutting is " << TT(0, n+1) << "." << endl;
39     }
40 }
41

```

## uva/10015.cpp

```

1  //10015
2  //Joseph's Cousin
3  //Dynamic Programming;Josephus Problem
4  #include <iostream>
5  #include <vector>
6  #define MAX 35000
7  #define MAX2 3503
8  using namespace std;
9
10 bool P[MAX];
11 vector<int> W;
12 int T[MAX2][MAX2];
13
14 int main() {
15     P[0] = P[1] = true;
16     for(int i=2; i<MAX; i++) {

```

```

17         if (!P[i]) {
18             W.push_back(i);
19             for(int j=i*i; j<MAX; j+=i)
20                 P[j] = true;
21         }
22     }
23
24     for(int i=2; i<MAX2; i++)
25         for(int j=0; j<MAX2-1; j++)
26             T[i][j] = (W[j] + T[i-1][j+1])%i;
27
28     int n;
29     while(cin >> n, n) {
30         cout << T[n][0]+1 << endl;
31     }
32
33 }

```

## uva/10018.cpp

```

1  //10018
2  //Reverse and Add
3  //Misc;Ad hoc
4  #include <iostream>
5  using namespace std;
6
7  long reverse(long a) {
8      long b = 0;
9      while(a) {
10         b = b*10 + a%10;
11         a /= 10;
12     }
13     return b;
14 }
15
16 int main() {
17     int n;
18     cin >> n;
19     while(n-->0) {
20         long a; cin >> a;
21         for(int i=1; i<=1000; i++) {
22             a = a + reverse(a);
23             if (a == reverse(a)) {
24                 cout << i << " " << a << endl;
25                 break;
26             }
27         }
28     }
29 }

```

## uva/10035.cpp

```

1  //10035
2  //Primary Arithmetic
3  //Misc;Ad hoc
4  #include <iostream>
5  using namespace std;
6
7  int main() {
8      int a, b;
9      while(cin >> a >> b, a|b) {
10         int carries = 0;
11         int c = 0;
12         while(a|b) {
13             int s = a%10 + b%10 + c;
14             c = s/10;
15             if (c) carries++;
16
17             a/=10;
18             b/=10;
19         }
20
21         if (carries == 0) {
22             cout << "No carry operation." << endl;
23         } else if (carries==1) {
24             cout << "1 carry operation." << endl;
25         } else {
26             cout << carries << " carry operations." << endl;
27         }
28     }
29 }

```

## uva/10044.cpp

```
1  //10044
2  //Erdos Number
3  //Graphs;Shortest Path;BFS
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <string>
8  #include <vector>
9  #include <queue>
10 #include <map>
11 #define MAX 5000
12 using namespace std;
13
14 vector<int> G[MAX];
15 int n, m;
16 bool V[MAX];
17 map<string, int> A;
18
19 struct Step {
20     int x, v;
21     Step() {}
22     Step(int x, int v) : x(x), v(v) {}
23 };
24
25 queue<Step> Q;
26
27 int author(const string& a) {
28     if (A.find(a) != A.end())
29         return A[a];
30     else
31         return A[a] = A.size()-1;
32 }
33
34
35 char C[MAX];
36 void parseAuthors(const string& s) {
37     vector<int> TA;
38     int commas = 0, chars=0;
39     for(int i=0;i<s.size();i++) {
40         char c = s[i];
41         if (chars == 0 && c == ' ') continue;
42
43         if ((c==',' || c==':') && ++commas == 2) {
44             TA.push_back(author(string(C, chars)));
45             chars = commas = 0;
46         } else {
47             C[chars++] = c;
48         }
49     }
50     for(int i=0;i<TA.size(); i++) {
51         for(int j=i+1;j<TA.size(); j++) {
52             G[TA[i]].push_back(TA[j]);
53             G[TA[j]].push_back(TA[i]);
54         }
55     }
56 }
57
58
59 int main() {
60     string s;
61     int t=0, tt;
62     cin >> tt;
63     while(t++ < tt) {
64         cin >> n >> m;
65         memset(G, 0, sizeof(G));
66         A.clear();
67         getline(cin, s);
68         while(n--) {
69             getline(cin, s);
70             parseAuthors(s);
71         }
72
73         cout << "Scenario " << t << endl;
74         for(int i=0;i<m;i++) {
75             bool stop;
76             memset(V, 0, sizeof(V));
77             getline(cin, s);
78             int b = author(s);
79             Q = queue<Step>();
80             Q.push(Step(author("Erdos, P."), 0));
81             bool found = false;
```

```

82         while(!Q.empty()) {
83             Step it = Q.front(); Q.pop();
84             if (it.x == b) {
85                 cout << s << " " << it.v << endl;
86                 found = true;
87                 break;
88             }
89             V[it.x] = true;
90             for(int i=0; i<G[it.x].size(); i++)
91                 if (!V[G[it.x][i]]) Q.push(Step(G[it.x][i], it.v+1));
92             if (!found) cout << s << " infinity" << endl;
93         }
94     }
95     return 0;
96 }
97
100 }

```

## uva/10051.cpp

```

1  //10051
2  //Tower of Cubes
3  //Dynamic Programming;Longest Increasing Subsequence
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #include <cmath>
8  #include <climits>
9  #define MAX 501
10 #define MAXC 101
11 using namespace std;
12
13 int T[MAX][MAXC], F[MAX][MAXC], P[MAX][MAXC];
14 int A[6];
15
16 string translate(int side) {
17     switch(side) {
18         case 0: return "front";
19         case 1: return "back";
20         case 2: return "left";
21         case 3: return "right";
22         case 4: return "top";
23         case 5: return "bottom";
24     }
25 }
26
27 void print(int first, int k) {
28     if (k==0) return;
29
30     print(F[k][first], k-1);
31     cout << T[k][first] << " " << translate(P[k][first]) << endl;
32 }
33
34 int main() {
35     int n, n2, t;
36     while(cin >> n, n) {
37         if (t++) cout << endl;
38         cout << "Case #" << t << endl;
39
40         memset(T, 0, sizeof(T));
41
42         for(int i=1; i<=MAXC; i++) {
43             T[0][i] = 1;
44         }
45         int k = 0;
46
47         for(int cube=1; cube<=n; cube++) {
48             for(int i=0; i<6; i++) cin >> A[i];
49             int newk = k;
50             for(int j=k; j>=0; j--) {
51                 for(int i=0; i<6; i++) {
52                     int other = (i/2*2)+(1-i%2);
53                     if (T[j][A[i]] && !T[j+1][A[other]]) {
54                         T[j+1][A[other]] = cube;
55                         F[j+1][A[other]] = A[i];
56                         P[j+1][A[other]] = i;
57                         newk = max(newk, j+1);
58                     }
59                 }
60             }
61             k=newk;

```

```

62     }
63
64     cout << k << endl;
65
66     int first=0;
67     for(int i=1;i<=100;i++)
68         if (T[k][i]) first=i;
69
70     print(first, k);
71 }
72
73 return 0;
74 }

```

## uva/10065.cpp

```

1  //10065
2  //Useless Tile Packers
3  //Math;Geometry;Convex Hull;Monotone Chain
4  #include <iostream>
5  #include <cmath>
6  #include <iomanip>
7  #include <algorithm>
8  using namespace std;
9
10 struct Point {
11     int x, y;
12
13     Point() {}
14     Point(int x, int y) : x(x), y(y) {}
15
16     bool left(Point& a, Point& b) {
17         return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x) < 0;
18     }
19
20     bool operator <(const Point& p) const {
21         if (this->x != p.x) return this->x < p.x;
22         return this->y < p.y;
23     }
24
25     bool operator ==(const Point& p) const {
26         return this->x == p.x and this->y == p.y;
27     }
28 };
29
30 double area(Point* A, int a) {
31     double area = 0;
32     for(int i=0; i<a; i++) {
33         int j = (i+1)%a;
34         area += (A[i].x + A[j].x) * (A[i].y - A[j].y);
35     }
36     return area / 2;
37 }
38
39 int convexHull(Point* P, int n, Point* S) {
40     sort(P, P+n);
41
42     int m=0;
43     for(int i=0; i<n; i++) {
44         while(m >= 2 && S[m-1].left(S[m-2], P[i])) m--;
45         S[m++] = P[i];
46     }
47     m--;
48
49     for(int i=n-1, k=m; i >= 0; i--) {
50         while(m >= k+2 && S[m-1].left(S[m-2], P[i])) m--;
51         S[m++] = P[i];
52     }
53     m--;
54
55     return m;
56 }
57
58 Point P[110], S[110];
59
60 int main() {
61     int n, tt=0;
62     while(cin >> n, n) {
63         for(int i=0; i<n; i++) {
64             int x, y; cin >> x >> y;
65             P[i] = Point(x, y);
66         }
67     }

```



```

68 |
69 |         double original = abs(area(P, n));
70 |
71 |         int m = convexHull(P, n, S);
72 |         double modified = abs(area(S, m));
73 |         double ratio = 100*(1.0-(original/modified));
74 |
75 |         cout << "Tile #" << ++tt << endl;
76 |         cout << "Wasted Space = " << fixed << setprecision(2) << ratio << " %" << endl;
77 |         cout << endl;
78 |     }
79 |
80 | }
81 |

```

## uva/10066.cpp

```

1 | //10066
2 | //The Twin Towers
3 | //Dynamic Programming;Longest Common Subsequence
4 | #include <iostream>
5 | #include <string>
6 | #include <cstring>
7 | #include <cmath>
8 | #define MAX 105
9 | using namespace std;
10 |
11 | int T[MAX][MAX];
12 | int P[MAX], Q[MAX];
13 |
14 | int main() {
15 |     int p, q, tt=0;
16 |     while(cin >> p >> q, tt++, p&&q) {
17 |         memset(T, 0, sizeof(T));
18 |
19 |         for(int i=0; i<p; i++) cin >> P[i];
20 |         for(int i=0; i<q; i++) cin >> Q[i];
21 |
22 |         for(int i=0; i<=p; i++) T[i][0] = 0;
23 |         for(int i=0; i<=q; i++) T[0][i] = 0;
24 |
25 |         for(int i=1; i<=p; i++) {
26 |             for(int j=1; j<=q; j++) {
27 |                 if (P[i-1] == Q[j-1])
28 |                     T[i][j] = T[i-1][j-1] + 1;
29 |                 else
30 |                     T[i][j] = max(T[i-1][j], T[i][j-1]);
31 |             }
32 |         }
33 |         cout << "Twin Towers #" << tt << endl;
34 |         cout << "Number of Tiles : " << T[p][q] << endl;
35 |         cout << endl;
36 |     }
37 |
38 |     return 0;
39 | }

```

## uva/10090.cpp

```

1 | //10090
2 | //Marbles
3 | //Math;Extended Euclid
4 | #include <iostream>
5 | #define ull long long
6 | using namespace std;
7 |
8 | ull euclid(ull a, ull b, ull& rx, ull& ry) {
9 |     if (!b) return rx=1, ry=0, a;
10 |
11 |     ull q = a/b;
12 |     ull x, y;
13 |     ull g = euclid(b, a-q*b, x, y);
14 |     return rx=y, ry=x-q*y, g;
15 | }
16 |
17 | ull solve(ull a, ull b, ull c) {
18 |     ull x, y;
19 |     ull g = euclid(a, b, x, y);
20 |     if (c%g) return -1;
21 |     ull ag=a/g, bg=b/g, cg=c/g;
22 |     return (x*cg%bg+bg)%bg;

```

```

23 }
24
25 int main() {
26     ull n, c1, n1, c2, n2;
27
28     while(cin >> n, n) {
29         cin >> c1 >> n1 >> c2 >> n2;
30
31         ull sol1=solve(n1, n2, n), sol2=solve(n2, n1, n);
32
33         ull sol12=(n-n1*sol1)/n2, sol21=(n-n2*sol2)/n1;
34
35         if (sol1 < 0 || sol12 < 0) {
36             cout << "failed" << endl;
37             continue;
38         }
39
40         ull cos1=c1*sol1+sol12*c2;
41         ull cos2=c2*sol2+sol21*c1;
42         if (cos1 < cos2)
43             cout << sol1 << " " << sol12 << endl;
44         else
45             cout << sol21 << " " << sol2 << endl;
46     }
47 }

```

## uva/10092.cpp

```

1 //10092
2 //The Problem with the Problem Setter
3 //Graphs;Maximum Flow;Ford-Fulkerson
4 #include <iostream>
5 #include <iomanip>
6 #include <cstring>
7 #include <string>
8 #include <cmath>
9 #include <climits>
10 #define MAX 1100
11 using namespace std;
12
13 int G[MAX][MAX], nk, np, n;
14 bool V[MAX];
15
16 int SOURCE() { return 1; }
17 int P(int i) { return 1+i; }
18 int K(int i) { return 1+np+i; }
19 int TARGET() { return 2+np+nk; }
20
21 int send(int s, int t, int minn) {
22     V[s] = true;
23
24     if (s==t) return minn;
25     for(int i=1; i<=n; i++) {
26         if (!V[i] && G[s][i] > 0) {
27             if (int sent = send(i, t, min(minn, G[s][i]))) {
28                 G[s][i] -= sent;
29                 G[i][s] += sent;
30                 return sent;
31             }
32         }
33     }
34     return 0;
35 }
36
37 int main() {
38     int tmp, tmp2;
39     while(cin >> nk >> np, nk|np) {
40         n = nk+np+2;
41         int expected = 0;
42         memset(G, 0, sizeof(G));
43         memset(V, 0, sizeof(V));
44
45         for(int i=1; i<=nk; i++) {
46             cin >> tmp;
47             expected += tmp;
48             G[K(i)][TARGET()] = tmp;
49         }
50
51         for(int i=1; i<=np; i++) {
52             cin >> tmp;
53             G[SOURCE()][P(i)] = 1;
54             for(int j=0; j<tmp; j++) {
55                 cin >> tmp2;

```

```

56         G[P(i)][K(tmp2)] = 1;
57     }
58 }
59
60 int total = 0;
61 while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
62     total += sent;
63     memset(V, 0, sizeof(V));
64 }
65 cout << (expected == total ? 1: 0) << endl;
66 if (expected == total) {
67     for(int i=K(1); i<=K(nk); i++) {
68         bool printed = false;
69         for(int j=P(1); j<=P(np); j++) {
70             if (G[i][j]) {
71                 cout << (printed?" ":"") << (j-1);
72                 printed = true;
73             }
74         }
75         cout << endl;
76     }
77 }
78 }
79 }

```

## uva/10104.cpp

```

1  //10104
2  //Euclid Problem
3  //Math;Extended Euclid
4  #include <iostream>
5  #define ull long long
6  using namespace std;
7
8  int euclid(int a, int b, int& rx, int& ry) {
9      if (!b) return rx=1, ry=0, a;
10
11     int q = a/b;
12     int x, y;
13     int g = euclid(b, a-q*b, x, y);
14     return rx=y, ry=x-q*y, g;
15 }
16
17 int main() {
18     int a, b;
19
20     while(cin >> a >> b) {
21         int x, y;
22         int d = euclid(a,b,x,y);
23
24         cout << x << " " << y << " " << d << endl;
25     }
26 }

```

## uva/10113.cpp

```

1  //10113
2  //Exchange Rates
3  //Graphs;DFS
4  #include <iostream>
5  #include <vector>
6  #include <map>
7  #include <set>
8  #include <algorithm>
9  #include <string>
10 using namespace std;
11
12 int gcd(int a, int b) {
13     while(b)
14         swap(a=a%b,b);
15     return a;
16 }
17
18 struct Edge {
19     string s;
20     int a, b;
21     Edge() : a(0), b(0) { }
22     Edge(string s, int a, int b) : s(s), a(a), b(b) {}
23
24     Edge next(Edge e) {
25         int na = a*e.a, nb = b*e.b;

```

```

26         int g = gcd(na, nb);
27         na /= g; nb /= g;
28         return Edge(e.s, na, nb);
29     }
30
31     bool valid() { return a!=0; }
32 };
33
34
35 map<string, vector<Edge> > G;
36 set<string> V;
37
38 Edge dfs(Edge e, string target) {
39     if (e.s == target) return e;
40     V.insert(e.s);
41     vector<Edge> ve = G[e.s];
42
43     for(int i=0; i<ve.size(); i++) {
44         if (V.find(ve[i].s) == V.end()) {
45             Edge other = dfs(e.next(ve[i]), target);
46             if (other.valid()) return other;
47         }
48     }
49
50     return Edge();
51 }
52
53 int main() {
54     string cmd;
55
56     while(cin >> cmd, cmd!=".") {
57         string s1, s2, temp;
58         if (cmd == "!") {
59             int x, y;
60             cin >> x >> s1 >> temp >> y >> s2;
61             G[s1].push_back(Edge(s2, x, y));
62             G[s2].push_back(Edge(s1, y, x));
63         } else if (cmd == "?") {
64             cin >> s1 >> temp >> s2;
65             V.clear();
66             Edge e = dfs(Edge(s1, 1, 1), s2);
67             if (e.valid())
68                 cout << e.a << " " << s1 << " = " << e.b << " " << s2 << endl;
69             else
70                 cout << "?" << s1 << " = ? " << s2 << endl;
71         }
72     }
73 }
74 }

```

## uva/10154.cpp

```

1  //10154
2  //Weights and Measures
3  //Dynamic Programming; Longest Increasing Subsequence
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #include <cmath>
8  #include <climits>
9  #include <vector>
10 #include <algorithm>
11 #define MAX 10005
12 using namespace std;
13
14 struct Turtle {
15     int w, c;
16     Turtle() {}
17     Turtle(int w, int c) : w(w), c(c) {}
18 };
19
20 bool compare(const Turtle& a, const Turtle& b) {
21     return a.c > b.c;
22 }
23
24 vector<Turtle> V;
25 int T[MAX];
26 int main() {
27     int w, c, k=0;
28     T[0] = INT_MAX;
29
30     while(cin >> w >> c) {
31         V.push_back(Turtle(w, c-w));

```

```

32     }
33     sort(V.begin(), V.end(), compare);
34
35     for(int i=0; i<V.size(); i++) {
36         int w = V[i].w, c = V[i].c;
37
38         for(int j=k; j>=0; j--) {
39             int next = min(T[j]-w, c);
40             if (next >= T[j+1]) {
41                 T[j+1] = next;
42                 k=max(k, j+1);
43             }
44         }
45     }
46     cout << k << endl;
47
48     return 0;
49 }

```

## uva/10158.cpp

```

1  //10158
2  //War
3  //Misc;Union-Find
4  #include <iostream>
5  #include <map>
6  #include <string>
7  #include <cstring>
8  #include <algorithm>
9  using namespace std;
10
11 int P[20000];
12
13 inline int enemy(int v) { return v+10000; }
14
15 inline int findset(int v) {
16     if (P[v] != -1 && P[v] != v)
17         return P[v] = findset(P[v]);
18     return v;
19 }
20
21 inline int unionset(int x, int y) {
22     int a = findset(x), b = findset(y);
23     if (a<b) swap(a,b);
24     P[b] = a;
25 }
26
27 int main() {
28     memset(P, -1, sizeof(P));
29     int n, c, x, y;
30     cin >> n;
31     while(cin >> c >> x >> y, c|x|y) {
32         if (c==1) {
33             if (findset(x) == findset(enemy(y))) { cout << -1 << endl; continue; }
34             unionset(x, y);
35             unionset(enemy(x), enemy(y));
36         } else if (c==2) {
37             if (findset(x) == findset(y)) { cout << -1 << endl; continue; }
38             unionset(x, enemy(y));
39             unionset(enemy(x), y);
40         } else if (c==3) {
41             cout << (findset(x) == findset(y)) << endl;
42         } else if (c==4) {
43             cout << (findset(x) == findset(enemy(y))) << endl;
44         }
45     }
46 }

```

## uva/10189.cpp

```

1  //10189
2  //Minesweeper
3  //Misc;Ad hoc
4  #include <iostream>
5  #include <cstring>
6  using namespace std;
7
8  char T[200][200];
9  int N[200][200];
10 int n, m, t=0;
11

```

```

12 void add(int i, int j) {
13     if (i<0 || i>=n || j<0 || j>=m) return;
14     N[i][j]++;
15 }
16
17 int main() {
18     while(cin >> n >> m, n|m) {
19         memset(N, 0, sizeof(N));
20         for(int i=0; i<n; i++) {
21             for(int j=0; j<m; j++) {
22                 cin >> T[i][j];
23                 if (T[i][j] == '*') {
24                     add(i-1, j-1);
25                     add(i-1, j);
26                     add(i-1, j+1);
27                     add(i, j-1);
28                     add(i, j+1);
29                     add(i+1, j-1);
30                     add(i+1, j);
31                     add(i+1, j+1);
32                 }
33             }
34         }
35
36         if (t++>0) cout << endl;
37         cout << "Field #" << t << ":" << endl;
38         for(int i=0; i<n; i++) {
39             for(int j=0; j<m; j++) {
40                 if (T[i][j] == '*')
41                     cout << T[i][j];
42                 else
43                     cout << N[i][j];
44             }
45             cout << endl;
46         }
47     }
48 }

```

## uva/10192.cpp

```

1 //10192
2 //Vacation
3 //Dynamic Programming;Longest Common Subsequence
4 #include <iostream>
5 #include <string>
6 #include <cstring>
7 #include <cmath>
8 #define MAX 1005
9 using namespace std;
10
11 int T[MAX][MAX];
12 string P, Q;
13
14 int main() {
15     int p, q, tt=0;
16     while(getline(cin, P), P!="#") {
17         tt++;
18         getline(cin, Q);
19         int p = P.size(), q = Q.size();
20
21         memset(T, 0, sizeof(T));
22
23         for(int i=0; i<=p; i++) T[i][0] = 0;
24         for(int i=0; i<=q; i++) T[0][i] = 0;
25
26         for(int i=1; i<=p; i++) {
27             for(int j=1; j<=q; j++) {
28                 if (P[i-1] == Q[j-1])
29                     T[i][j] = T[i-1][j-1] + 1;
30                 else
31                     T[i][j] = max(T[i-1][j], T[i][j-1]);
32             }
33         }
34         cout << "Case #" << tt << ": you can visit at most " << T[p][q] << " cities." << endl;
35     }
36
37     return 0;
38 }

```

## uva/10199.cpp

```

1 //10199
2 //Tourist Guide
3 //Graphs;Finding Articulation Points
4 #include <iostream>
5 #include <cstring>
6 #include <map>
7 #include <vector>
8 #include <algorithm>
9 #define MAX 1001
10 using namespace std;
11 int G[MAX][MAX], V[MAX], L[MAX], P[MAX], n, m, gpe;
12 map<string, int> S;
13 string SR[MAX];
14 vector<string> F;
15
16 void dfs(int u, int v) {
17     V[v] = L[v] = ++gpe;
18     for(int i = 1; i <= n; i++) {
19         if(G[v][i]) {
20             if(!V[i]) {
21                 dfs(v, i);
22                 L[v] = min(L[v], L[i]);
23                 if(L[i] >= V[v])
24                     P[v]++;
25             } else if(i != u) {
26                 L[v] = min(L[v], V[i]);
27             }
28         }
29     }
30 }
31
32 int main() {
33     int tt = 0;
34     while(cin >> n, n) {
35         memset(G, 0, sizeof(G));
36         memset(V, 0, sizeof(V));
37         memset(L, 0, sizeof(L));
38         memset(P, 0, sizeof(P));
39         S.clear();
40         F.clear();
41         gpe = 0;
42
43         for(int i=1; i<=n; i++) {
44             string s; cin >> s;
45             S[s] = i;
46             SR[i] = s;
47         }
48
49         cin >> m;
50         for(int i=0; i<m; i++) {
51             string s1, s2; cin >> s1 >> s2;
52             int a = S[s1], b = S[s2];
53             G[a][b] = G[b][a] = 1;
54         }
55
56         for(int i=1; i<=n; i++) {
57             if (!V[i]) {
58                 dfs(i, i);
59                 P[i]--;
60             }
61         }
62
63         for(int i=1; i<=n; i++)
64             if (P[i]>0)
65                 F.push_back(SR[i]);
66
67         sort(F.begin(), F.end());
68
69         if (tt) cout << endl;
70
71         cout << "City map #" << ++tt << ": " << F.size() << " camera(s) found" << endl;
72         for(int i=0; i<F.size(); i++)
73             cout << F[i] << endl;
74     }
75 }

```

## uva/10243.cpp

```

1 //10243
2 //Fire! Fire! Fire!
3 //Graphs;DFS
4 #include <iostream>
5 #include <iomanip>

```

```

6  #include <cstring>
7  #include <queue>
8  #include <cmath>
9  #define MAX 1005
10 using namespace std;
11
12 int n;
13 bool G[MAX][MAX];
14 bool L[MAX];
15 bool V[MAX];
16
17 void dfs(int v, bool start) {
18     //cout << "*" << v << endl;
19     if (V[v]) return;
20     V[v] = true;
21     bool all = true;
22     int children = 0;
23     for(int i=0;i<n;i++) {
24         if (G[v][i] && !V[i]) {
25             dfs(i, false);
26             all &= L[i];
27             children++;
28         }
29     }
30     if (!all && children > 0 || start && children==0)
31         L[v] = true;
32 }
33
34 int main() {
35     int a, b, m;
36
37     while(cin >> n, n) {
38         memset(G, 0, sizeof(G));
39         memset(L, 0, sizeof(L));
40         memset(V, 0, sizeof(V));
41
42         for(int i=0;i<n;i++) {
43             cin >> m;
44             for(int j=0; j<m; j++) {
45                 cin >> a;
46                 a--;
47                 G[i][a] = G[a][i] = true;
48             }
49         }
50
51         int count = 0;
52         for(int i=0;i<n;i++) {
53             dfs(i, true);
54             if (L[i]) count++;
55         }
56
57         cout << count << endl;
58     }
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-->0) {
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-->0) {
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++) {
26             int a; cin >> a; F[a] = true;
27         }
28         getline(cin, s);
29         while(getline(cin, s), cin && s!="") {
30             int a, b, c;
31             stringstream inter(s);
32             inter >> a >> b >> c;
33             G[a][b] = G[b][a] = c;
34         }
35
36
37         for(int k=1; k<=n; k++) {
38             G[k][k] = 0;
39             for(int i=1; i<=n; i++)
40                 for(int j=1; j<=n; j++)
41                     G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
42         }
43
44         int minn = INT_MAX, minv;
45         for(int i=1; i<=n; i++) {
46             int maxx = 0;
47             for(int j=1; j<=n; j++) {
48                 int nearest = INT_MAX;
49                 for(int k=1; k<=n; k++) {
50                     if (!F[k] && k!=i) continue;
51                     nearest = min(nearest, G[k][j]);
52                 }
53                 maxx = max(maxx, nearest);
54             }
55             if (maxx < minn) {
56                 minn = maxx;
57                 minv = i;
58             }
59         }

```

```

60         cout << minv << endl;
61         if (t) cout << endl;
62     }
63 }

```

## uva/10298.cpp

```

1  //10298
2  //Power Strings
3  //Misc;String Matching;KMP
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #define MAX 1000010
8  using namespace std;
9
10 int F[MAX];
11
12 void kmp_init(string& P) {
13     F[0] = 0; F[1] = 0;
14     int i = 1, j = 0;
15     while(i < P.size()) {
16         if (P[i] == P[j])
17             F[++i] = ++j;
18         else if (j == 0)
19             F[++i] = 0;
20         else
21             j = F[j];
22     }
23 }
24
25 int kmp(string& P, string& T, int start) {
26     kmp_init(P);
27     int i = start, j = 0;
28     int n = T.size(), m = P.size();
29
30     while(i-j <= n-m) {
31         while(j < m) {
32             if (P[j] == T[i]) {
33                 i++; j++;
34             } else break;
35         }
36         if (j == m) return i-m;
37         else if (j == 0) i++;
38         j = F[j];
39     }
40 }
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
3 //Math;Geometry;Great-Circle Distance
4 #define PI 3.14159265
5 #include <iostream>
6 #include <cmath>
7 #include <iomanip>
8 #include <algorithm>
9 using namespace std;
10
11 struct Point {
12     double x, y;
13     double dx, dy;
14
15     Point() {}
16     Point(double x, double y) : x(x), y(y) {
17         dx = x/180.0*PI;
18         dy = y/180.0*PI;
19     }
20
21     double distance(Point& p) {
22         return acos(cos(p.dx) * cos(this->dx) * cos(this->dy - p.dy) + sin(p.dx) * sin(this->dx));
23     }
24 };
25
26 Point P[1050];
27
28 int main() {
29     int n;
30     while(cin >> n) {
31         for(int i=0; i<n; i++) {
32             double x,y; cin >> x >> y;
33             P[i] = Point(x,y);
34         }
35
36         double minn = 1000000000;
37         int mini = 0;
38         for(int i=0; i<n; i++) {
39             double maxx = 0;
```

```

40         for(int j=0;j<n;j++)
41             maxx = max(maxxx, P[i].distance(P[j]));
42
43         if (maxxx < minn || abs(maxxx - minn) < 1e-6) {
44             mini = i;
45             minn = maxxx;
46         }
47     }
48
49     cout << fixed << setprecision(2) << P[mini].x << " " << P[mini].y << endl;
50 }
51
52 }
53

```

## uva/10319.cpp

```

1  //10319
2  //Manhattan
3  //Graphs;2-SAT
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #include <vector>
8  #define MAX 1000
9  using namespace std;
10
11 vector<int> G[MAX*2], T[MAX*2];
12 int O[MAX*2], V[MAX*2], npv, n, s, a;
13
14 int neg(int x) {
15     if (x>=n) return x-n;
16     return x+n;
17 }
18
19 int av(int x) {
20     return s+x;
21 }
22
23 int st(int x) {
24     return x;
25 }
26
27 void DFS(int v){
28     V[v] = 1;
29     for(int i = 0; i < G[v].size(); i++)
30         if (!V[G[v][i]])
31             DFS(G[v][i]);
32     O[npv++] = v;
33 }
34
35 void DFSt(int v, int comp){
36     V[v] = comp;
37     for(int i = 0; i < T[v].size(); i++)
38         if (!V[T[v][i]])
39             DFSt(T[v][i], comp);
40 }
41
42
43 int main() {
44     int m;
45     int tt; cin >> tt;
46     while(tt--){
47         cin >> s >> a >> m;
48         n = (s+a);
49
50         for(int i=0;i<2*n; i++) {
51             G[i].clear();
52             T[i].clear();
53         }
54
55         for(int i=0; i<m; i++) {
56             int s1, a1, s2, a2;
57             cin >> s1 >> a1 >> s2 >> a2;
58             s1--; a1--; s2--; a2--;
59
60             s1 = st(s1); s2=st(s2);
61             a1 = av(a1); a2=av(a2);
62
63             if (a1 == a2 && s1 == s2)
64                 continue;
65
66             if (a2<a1) {

```

```

67         s1 = neg(s1);
68         s2 = neg(s2);
69     }
70
71     if (s2 < s1) {
72         a1 = neg(a1);
73         a2 = neg(a2);
74     }
75
76     if (a1 == a2) {
77         G[neg(a1)].push_back(a1);
78         continue;
79     }
80
81     if (s1 == s2) {
82         G[neg(s1)].push_back(s1);
83         continue;
84     }
85
86     G[neg(s1)].push_back(a1);
87     G[neg(a1)].push_back(s1);
88
89     G[neg(s1)].push_back(s2);
90     G[neg(s2)].push_back(s1);
91
92     G[neg(a2)].push_back(a1);
93     G[neg(a1)].push_back(a2);
94
95     G[neg(a2)].push_back(s2);
96     G[neg(s2)].push_back(a2);
97 }
98
99 for(int i=0; i<2*n; i++)
100     for(int j=0; j<G[i].size(); j++)
101         T[G[i][j]].push_back(i);
102
103
104 npv = 0;
105 memset(V, 0, sizeof(V));
106 memset(O, 0, sizeof(O));
107
108 for(int i = 0; i < 2*n; i++)
109     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[0[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=t-1;
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++) {
63                     double aDist = dist(X[n], Y[n], X[i], Y[i])/WALK;
64                     G[i].push_back(Edge(n, aDist));
65                     G[n].push_back(Edge(i, aDist));
66                 }
67                 n++; mn++;
68             }
69         }
70
71         int totalc=0;
72
73         for(int i=0; i<n; i++) V[i] = -1;
74
75         priority_queue<Edge> Q;
76         Q.push(Edge(0, 0));
77
78         while(totalc < n && !Q.empty()) {
79             Edge item = Q.top(); Q.pop();
80             if (item.c >= V[item.v] && V[item.v] >= 0) continue;
81
82             V[item.v] = item.c;
83             totalc++;
84
85             for(int j=0; j<G[item.v].size(); j++) {
86                 Edge e = G[item.v][j];
87                 if (item.c + e.c < V[e.v] || V[e.v] == -1)
88                     Q.push(Edge(e.v, item.c + e.c));
89             }
90         }
91
92         if (t++) cout << endl;
93         cout << (int)round(V[1]) << endl;
94     }
95     return 0;
96 }
97
98 }

```

## uva/10397.cpp

```

1 //10397
2 //Connect the Campus
3 //Graphs;Minimum Spanning Tree;Prim;Priority Queue

```

```

4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <vector>
8  #include <algorithm>
9  #include <queue>
10 #include <cmath>
11 #include <iomanip>
12 #define MAX 200010
13
14 using namespace std;
15
16 struct Road {
17     int v; double c;
18     Road(int v, double c) : v(v), c(c) {}
19     inline bool operator < (const Road& that) const { return c > that.c; }
20 };
21
22
23 vector<Road> G[MAX];
24 int X[MAX], Y[MAX];
25 priority_queue<Road> Q;
26 int n, m;
27 bool V[MAX];
28
29
30 int main() {
31     while(cin >> n) {
32         memset(V, 0, sizeof(V));
33         memset(G, 0, sizeof(G));
34         Q = priority_queue<Road>();
35
36         for(int i=1; i<=n; i++) {
37             int x, y;
38             cin >> x >> y;
39             X[i] = x; Y[i] = y;
40             for(int j=1; j<=i; j++) {
41                 double d = sqrt(pow(X[i]-X[j], 2.0)+pow(Y[i]-Y[j], 2.0));
42                 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-->0) {
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];
13 | int main() {
14 |     int n=201, p=21, t=0;
15 |
16 |     for(int i=0; i<=n; i++) {
17 |         if (i<31)
18 |             T[i][3] = (1<<i)-1;
19 |         else
20 |             T[i][3] = INT_MAX; //avoid overflow
21 |     }
22 |
23 |     for(int i=1; i<=n; i++) {
24 |         for(int j=4; j<=p; j++) {
25 |             if (i<j) {
26 |                 T[i][j] = 2*i-1;
27 |             } else {
28 |                 int minn = INT_MAX;
29 |                 for(int k=1; k<i; k++) {
30 |                     int value = 2*T[k][j]+T[i-k][j-1];
31 |                     if (value >= 0) //avoid overflow
32 |                         minn = min(minn, value);
33 |                 }
34 |                 T[i][j] = minn;
35 |             }
36 |         }
37 |     }
38 |
39 |
40 |     while(cin >> n >> p, n | p) {
41 |         cout << "Case " << ++t << ": " << T[n][p] << endl;
42 |     }
43 |
44 |     return 0;
45 | }

```

## uva/10462.cpp

```

1 | //10462
2 | //Is There A Second Way Left?
3 | //Graphs;Minimum Spanning Tree;Kruskal
4 | #include <iostream>
5 | #include <cstring>
6 | #include <vector>
7 | #include <algorithm>
8 | #include <cassert>
9 | using namespace std;
10 |

```



```

11 struct Edge {
12     int x, y, v;
13     inline bool operator <(const Edge& that) const {
14         return this->v < that.v;
15     }
16 };
17
18 Edge E[205];
19 int P[105], S[105];
20
21 inline int findset(int v) {
22     if (P[v] != v)
23         return P[v] = findset(P[v]);
24     return v;
25 }
26
27 inline int unionset(int x, int y) {
28     int a = findset(x), b = findset(y);
29     if (a==b) return -1;
30     if (a>b) swap(a,b);
31     P[b] = a;
32     return a;
33 }
34
35 int best(int n, int m, int skip) {
36     for(int i=0; i<=n; i++)
37         P[i] = i;
38
39     int total=0, count=0;
40     for(int i=0; i<m && count < n-1; i++) {
41         if(i!=skip && unionset(E[i].x, E[i].y) != -1) {
42             total += E[i].v;
43             if (skip == -1)
44                 S[count] = i;
45             count++;
46         }
47     }
48     if (count == n-1)
49         return total;
50     else
51         return -1;
52 }
53
54 int main() {
55     int t; cin >> t;
56     for(int tt=1; tt<=t; tt++) {
57         int n, m; cin >> n >> m;
58
59         for(int i=0; i<m; i++)
60             cin >> E[i].x >> E[i].y >> E[i].v;
61
62         sort(E, E+m);
63
64         cout << "Case #" << tt << " : ";
65
66         if (best(n, m, -1) == -1) {
67             cout << "No way" << endl;
68             continue;
69         }
70
71         int minn = 1<<30;
72         for(int i=0; i<n-1; i++) {
73             int value = best(n, m, S[i]);
74             if (value != -1)
75                 minn = min(minn, value);
76         }
77
78         if (minn < 1<<30)
79             cout << minn << endl;
80         else
81             cout << "No second way" << endl;
82     }
83 }
84

```

## uva/10480.cpp

```

1 //10480
2 //Sabotage
3 //Graphs;Maximum Flow;Ford-Fulkerson
4 #include <iostream>
5 #include <iomanip>
6 #include <cstring>

```

```

7  #include <string>
8  #include <cmath>
9  #include <climits>
10 #define MAX 1006
11 using namespace std;
12
13 int G[MAX][MAX], O[MAX][MAX], n, m;
14 bool V[MAX];
15
16 int send(int s, int t, int minn) {
17     V[s] = true;
18
19     if (s==t) return minn;
20     for(int i=1; i<=n; i++) {
21         if (!V[i] && G[s][i] > 0) {
22             if (int sent = send(i, t, min(minn, G[s][i]))) {
23                 G[s][i] -= sent;
24                 G[i][s] += sent;
25                 return sent;
26             }
27         }
28     }
29     return 0;
30 }
31
32 int main() {
33     int tt=0;
34     while(cin >> n >> m, n|m) {
35         if (tt++) cout << endl;
36
37         memset(G, 0, sizeof(G));
38         memset(V, 0, sizeof(V));
39         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  //Counselling
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     if (s==t) return minn;
28     for(int i=1; i<=n; i++) {
29         if (!V[i] && G[s][i] > 0) {
30             if (int sent = send(i, t, min(minn, G[s][i]))) {
31                 G[s][i] -= sent;
32                 G[i][s] += sent;
33                 return sent;
34             }
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;
58             }
59         }
60
61         int maxParty = (EC.size()-1)/2;
62         for(map<string, int>::iterator it=EP.begin(); it!=EP.end(); it++) {
63             G[SOURCE()][it->second] = maxParty;
64         }
65
66         int total = 0;
67         while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
68             total += sent;
69             memset(V, 0, sizeof(V));
70         }
71
72         if (total == EC.size()) {
73             for(map<string, int>::iterator i=EM.begin(); i!=EM.end(); i++) {
74                 for(map<string, int>::iterator j=EC.begin(); j!=EC.end(); j++) {
75                     if (G[j->second][i->second]) {
76                         cout << i->first << " " << j->first << endl;
77                     }
78                 }
79             }
80         }
81         else {
82             cout << "Impossible." << endl;
83         }
84
85         if (t) cout << endl;
86     }
87 }

```

## uva/10557.cpp

```

1 //10557
2 //XYZZY
3 //Graphs;Shortest Path;Bellman Ford
4 #include <iostream>
5 #include <cstring>
6 #include <vector>
7 #include <algorithm>
8 #define MAX 1001
9 using namespace std;
10
11 struct Edge {
12     int a, b;
13     Edge(int a, int b) : a(a), b(b) {}
14 };
15
16 int X[MAX], M[MAX], V[MAX], n;
17 vector<Edge> E;
18
19 bool reach(int v, int t) {

```

```

20     if (v==t) return true;
21     V[v] = true;
22     for(int i=0; i<E.size(); i++) {
23         Edge e = E[i];
24         if (e.a == v && !V[e.b] && reach(e.b, t))
25             return true;
26     }
27     return false;
28 }
29
30 int main() {
31     while(cin >> n, n!=-1) {
32         memset(V, 0, sizeof(V));
33         E.clear();
34         for(int a=1; a<=n; a++) {
35             int k; cin >> X[a] >> k;
36
37             for(int j=0; j<k; j++) {
38                 int b; cin >> b;
39                 E.push_back(Edge(a, b));
40             }
41         }
42
43         M[1] = 100;
44         for(int a=2; a<=n; a++)
45             M[a] = -1<<29;
46
47         for(int k=0; k<n-1; k++) {
48             for(int i=0; i<E.size(); i++) {
49                 Edge e = E[i];
50                 if (M[e.a]<=0) continue;
51                 M[e.b] = max(M[e.b], M[e.a] + X[e.b]);
52             }
53         }
54
55         bool cycle = false;
56         for(int i=0; i<E.size(); i++) {
57             Edge e = E[i];
58             if (M[e.a]<=0) continue;
59             cycle |= M[e.a]+X[e.b] > M[e.b] && reach(e.a, n);
60         }
61
62         if (M[n] > 0 || cycle) {
63             cout << "winnable" << endl;
64         } else {
65             cout << "hopeless" << endl;
66         }
67     }
68 }
69 }
70 }

```

## uva/10594.cpp

```

1  //10594
2  //Data Flow
3  //Graphs;Maximum Flow;Min Cost;Cycle Canceling
4  #include <iostream>
5  #include <iomanip>
6  #include <cstring>
7  #include <string>
8  #include <cmath>
9  #define MAX 110
10 using namespace std;
11
12 struct Item {
13     long long v, p, c;
14     Item() {}
15     Item(long long v, long long p, long long c) : v(v), p(p), c(c) {}
16 };
17
18 long long C[MAX][MAX], M[MAX];
19 long long F[MAX][MAX], G[MAX][MAX], P[MAX], n, a, b, qn;
20 bool V[MAX];
21 Item Q[MAX];
22
23
24 long long send(long long s, long long t) {
25     memset(V, 0, sizeof(V));
26     qn = 0;
27
28     Q[qn++] = Item(s, -1, 1<<60);
29     V[s] = true;

```

```

30
31     for(long long i=0; i<qn; i++) {
32         Item item = Q[i];
33
34         if (item.v == t) {
35             long long sent = item.c;
36             while(item.p != -1) {
37                 Item parent = Q[item.p];
38                 F[parent.v][item.v] += sent;
39                 F[item.v][parent.v] -= sent;
40                 item = parent;
41             }
42
43             return sent;
44         }
45
46         for(long long j=0; j<n; j++) {
47             long long residual = G[item.v][j]-F[item.v][j];
48             if (!V[j] && residual) {
49                 V[j] = true;
50                 Q[qn++] = Item(j, i, min(item.c, residual));
51             }
52         }
53     }
54
55     return 0;
56 }
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
139     long long cost = 0;
140     for(long long i=0; i<n; i++)
141         for(long long j=0; j<n; j++)
142             if (F[i][j] > 0) {
143                 // cout << " " << i << " " << j << " " << F[i][j] << " " << C[i][j] << endl;
144                 cost += F[i][j] * C[i][j];
145             } else if (F[i][j] < 0) {
146                 // cout << " " << i << " " << j << " " << F[i][j] << " " << C[i][j] << endl;
147             }
148
149     if (total != d)
150         cout << "Impossible." << endl;
151     else
152         cout << cost << endl;
153 }
154 }
155 }

```

## uva/10635.cpp

```

1  //10635
2  //Prince and Princess
3  //Dynamic Programming;Longest Increasing Subsequence
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <cmath>
8  #include <algorithm>
9  #define MAX 255*255
10 using namespace std;
11
12 int P[MAX], Q[MAX], M[MAX];
13
14 int main() {
15     int n, p, q, tt=0, temp;
16     cin >> n;
17     while(cin >> n >> p >> q) {
18         memset(P, 0, sizeof(P));
19         q++; p++;
20         for(int i=1; i<=p; i++) {
21             cin >> temp;
22             P[temp] = i;
23         }
24
25         for (int i=1; i<=q; i++) {
26             cin >> temp;
27             Q[i] = P[temp];
28         }
29
30         int k=0; M[0]=0;
31         for(int i=1; i<=q; i++) {
32             if (Q[i] > M[k]) {
33                 k++; M[k] = Q[i];
34             } else {
35                 int j = (int)(lower_bound(M, M+k+1, Q[i]) - M);
36                 if (Q[i] > M[j]) j++;
37                 M[j] = Q[i];
38             }
39         }
40     }
41 }

```

```

39     }
40
41     cout << "Case " << ++tt << ": " << k << endl;
42 }
43
44     return 0;
45 }

```

## uva/10652.cpp

```

1  //10652
2  //Board Wrapping
3  //Math;Geometry;Convex Hull;Monotone Chain
4  #include <iostream>
5  #include <cmath>
6  #include <iomanip>
7  #include <algorithm>
8  using namespace std;
9
10 double PI = 2*acos(0.0);
11
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-->0) {
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);
94             Point b = Point(p.x-w/2, p.y+h/2).rotateWith(p, sin(angle), cos(angle), 1);
95             Point c = Point(p.x+w/2, p.y+h/2).rotateWith(p, sin(angle), cos(angle), 1);
96             Point d = Point(p.x+w/2, p.y-h/2).rotateWith(p, sin(angle), cos(angle), 1);
97
98             P[i*4+0] = a;
99             P[i*4+1] = b;
100            P[i*4+2] = c;
101            P[i*4+3] = d;
102
103            initialArea += area(P+i*4, 4);
104        }
105
106        int s = convexHull(P, n*4, S);
107        double finalArea = area(S, s);
108
109        cout << fixed << setprecision(1) << abs(100*(initialArea/finalArea)) << " %" << endl;
110    }
111 }

```

## uva/10684.cpp

```

1  //10684
2  //The Jackpot
3  //Dynamic Programming;Maximum Sum Contiguous Subsequence
4  #include <iostream>
5  #include <cmath>
6  #define MAX 1005
7  using namespace std;
8
9  int main() {
10     int n, a;
11     while(cin >> n, n) {
12         int t=0, s=0;
13         for(int i=0; i<n; i++) {
14             cin >> a;
15             if (s+a>=0)
16                 t = max(t, s+a);
17             else
18                 s = 0;
19         }
20         if (s>0) {
21             cout << "The maximum winning streak is " << t << "." << endl;
22         } else {
23             cout << "Losing streak." << endl;
24         }
25     }
26
27     return 0;
28 }

```

## uva/10696.cpp

```

1  //10694
2  //f91
3  //Misc;Ad hoc
4  #include <iostream>
5  using namespace std;
6
7  int f91(int x) {
8      if (x<=100)
9          return f91(f91(x+11));

```



```

10     else
11         return x-10;
12     }
13
14     int main() {
15         int n;
16         while(cin >> n, n)
17             cout << "f91(" << n << ") = " << f91(n) << endl;
18     }

```

## uva/10723.cpp

```

1 //10723
2 //Cyborg Genes
3 //Dynamic Programming;Longest Common Subsequence
4 #include <iostream>
5 #include <string>
6 #include <cstring>
7 #include <cmath>
8 #define MAX 1005
9 using namespace std;
10
11 int T[MAX][MAX], D[MAX][MAX];
12 string P, Q;
13
14 int main() {
15     int p, q, t, tt=0;
16     cin >> t;
17     getline(cin, P);
18     while(tt++ < t) {
19         getline(cin, P);
20         getline(cin, Q);
21         int p = P.size(), q = Q.size();
22
23         for(int i=0; i<=p; i++) { T[i][0] = 0; D[i][0] = 1; }
24         for(int i=0; i<=q; i++) { T[0][i] = 0; D[0][i] = 1; }
25
26         for(int i=1; i<=p; i++) {
27             for(int j=1; j<=q; j++) {
28                 D[i][j] = 0;
29                 if (P[i-1] == Q[j-1]) {
30                     T[i][j] = T[i-1][j-1] + 1;
31                     D[i][j] = D[i-1][j-1];
32                 }
33                 else {
34                     T[i][j] = max(T[i-1][j], T[i][j-1]);
35                     if (T[i-1][j] == T[i][j]) D[i][j] += D[i-1][j];
36                     if (T[i][j-1] == T[i][j]) D[i][j] += D[i][j-1];
37                 }
38             }
39         }
40
41         cout << "Case #" << tt << ": " << p+q-T[p][q] << " " << D[p][q] << endl;
42     }
43
44     return 0;
45 }

```

## uva/10724.cpp

```

1 //10724
2 //Road Construction
3 //Graphs;Shortest Path;Floyd-Warshall
4 #include <iostream>
5 #include <cmath>
6 #define MAX 58
7 #define EP 1e-6
8 #define eq(x, y) abs(x-y) < EP
9 using namespace std;
10
11 double G[MAX][MAX], X[MAX], Y[MAX];
12
13 double dist(int i, int j) {
14     return sqrt(pow(X[i]-X[j], 2.0) + pow(Y[i]-Y[j], 2.0));
15 }
16
17 struct Answer {
18     double x, d;
19     int i, j;
20     Answer(double x, double d, int i, int j) : x(x), d(d), i(i), j(j) {}
21 }

```

```

22     bool operator <(const Answer& o) const{
23         if (leq(x, o.x)) return x < o.x;
24         if (leq(d, o.d)) return d > o.d;
25         if (i!=o.i) return i>o.i;
26         if (j!=o.j) return j>o.j;
27         return false;
28     }
29
30     bool valid() {
31         return x-1 > EP;
32     }
33 };
34
35 int main() {
36     int n, m;
37     while(cin >> n >> m, n|m) {
38         for(int i=1; i<=n; i++) {
39             for(int j=1; j<=n; j++)
40                 G[i][j] = 1e8;
41
42             G[i][i] = 0;
43             cin >> X[i] >> Y[i];
44         }
45
46         for(int i=0; i<m; i++) {
47             int x, y;
48             cin >> x >> y;
49             G[x][y] = G[y][x] = dist(x, y);
50         }
51
52         for(int k=1; k<=n; k++)
53             for(int i=1; i<=n; i++)
54                 for(int j=1; j<=n; j++)
55                     G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
56
57         Answer maxx(0, 0, 0, 0);
58
59         for(int u=1; u<=n; u++) {
60             for(int v=1; v<=n; v++) {
61                 double improve = 0, uv = dist(u, v);
62
63                 for(int i=1; i<=n; i++)
64                     for(int j=1; j<=n; j++)
65                         improve += G[i][j] - min(G[i][j],
66                                                     min(G[i][u]+uv+G[v][j], G[i][v]+uv+G[u][j]));
67
68                 maxx = max(maxx, Answer(improve, uv, u, v));
69             }
70         }
71
72         if (maxx.valid()) {
73             cout << maxx.i << " " << maxx.j << endl;
74         } else {
75             cout << "No road required" << endl;
76         }
77     }
78 }

```

## uva/10739.cpp

```

1  //10739
2  //String to Palindrome
3  //Dynamic Programming;Edit Distance
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #include <cmath>
8  #define MAX 1005
9  using namespace std;
10
11 int T[MAX][MAX];
12 string P, Q;
13
14 int main() {
15     int p, q, t, tt=0;
16     cin >> t;
17     getline(cin, P);
18     while(tt++ < t) {
19         getline(cin, P);
20         Q = string(P.rbegin(), P.rend());
21         int p = P.size(), q = Q.size();
22
23         for(int i=0; i<=p; i++) { T[i][0] = i; }

```

```

24     for(int i=0; i<=q; i++) { T[0][i] = i; }
25
26     for(int i=1; i<=p; i++) {
27         for(int j=1; j<=q; j++) {
28             if (P[i-1] == Q[j-1])
29                 T[i][j] = T[i-1][j-1];
30             else
31                 T[i][j] = min(min(T[i-1][j], T[i][j-1]), T[i-1][j-1])+1;
32         }
33     }
34
35     cout << "Case " << tt << ": " << T[p][q]/2 << endl;
36 }
37
38 return 0;
39 }

```

## uva/10746.cpp

```

1  //10746
2  //Crime Wave - The Sequel
3  //Graphs;Maximum Flow;Min Cost;Cycle Canceling
4  #include <iostream>
5  #include <iomanip>
6  #include <cstring>
7  #include <string>
8  #include <cmath>
9  #define MAX 100
10 using namespace std;
11
12 double C[MAX][MAX], M[MAX];
13 int F[MAX][MAX], G[MAX][MAX], P[MAX], n, a, b;
14 bool V[MAX];
15
16 int send(int s, int t, int minn) {
17     V[s] = true;
18
19     if (s==t) return minn;
20
21     for(int i=0; i<n; i++) {
22         if (!V[i] && G[s][i]-F[s][i]) {
23             if (int sent = send(i, t, min(minn, G[s][i]-F[s][i]))) {
24                 F[s][i] += sent;
25                 F[i][s] -= sent;
26                 return sent;
27             }
28         }
29     }
30
31     return 0;
32 }
33
34 bool cancel_cycle(int source) {
35     memset(M, 0x1f, sizeof M);
36     M[source] = 0;
37
38     bool cycle = false;
39     int v;
40
41     for(int k=0; k<n; k++)
42     for(int i=0; i<n; i++)
43     for(int j=0; j<n; j++)
44         if (G[i][j]-F[i][j] && M[i]+C[i][j]<M[j]) {
45             cycle = k+1==n;
46             v = i;
47
48             M[j] = M[i] + C[i][j];
49             P[j] = i;
50         }
51
52     if (not cycle) return false;
53
54     for(int k=0; k<n; k++)
55         v = P[v];
56
57     int i=v, minn=1<<29;
58
59     do {
60         minn = min(minn, G[P[i]][i] - F[P[i]][i]);
61     } while (i = P[i], i!=v);
62
63     do {
64         F[P[i]][i] += minn;

```

```

65     F[i][P[i]] -= minn;
66     } while (i = P[i], i!=v);
67
68     return true;
69 }
70
71 int cruiser(int x) { return x; }
72 int bank(int x) { return b+x; }
73 int source() { return a+b; }
74 int target() { return a+b+1; }
75
76 int main() {
77     while(cin >> a >> b, a|b) {
78         memset(G, 0, sizeof(G));
79         memset(F, 0, sizeof(F));
80         memset(C, 0, sizeof(C));
81
82         for(int i=0; i<a; i++)
83             G[bank(i)][target()] = 1;
84
85         for(int i=0; i<b; i++)
86             G[source()][cruiser(i)] = 1;
87
88         for(int i=0; i<a; i++) {
89             for(int j=0; j<b; j++) {
90                 int cr = cruiser(j), bk = bank(i);
91                 cin >> C[cr][bk];
92                 G[cr][bk] = 1;
93                 C[bk][cr] = -C[cr][bk];
94             }
95         }
96
97         n = target()+1;
98
99         int total = 0, sent;
100         while(memset(V, 0, sizeof V), sent = send(source(), target(), 1<<29))
101             total += sent;
102
103         while(cancel_cycle(source()));
104
105         double cost = 0;
106         for(int i=0; i<n; i++)
107             for(int j=0; j<n; j++)
108                 if (F[i][j] > 0)
109                     cost += F[i][j] * C[i][j];
110
111         cout << fixed << setprecision(2) << cost/a+1e-6 << endl;
112     }
113 }

```

## uva/10783.cpp

```

1 //10783
2 //Odd Sum
3 //Misc;Ad hoc
4 #include <iostream>
5 #include <cstring>
6 #include <cmath>
7 using namespace std;
8
9 int main() {
10     int t;
11     cin >> t;
12     for(int tt=1;tt<=t;tt++) {
13         int a, b;
14         cin >> a >> b;
15         int s = 0;
16         for(int i=a;i<=b;i++) {
17             if (i&1)
18                 s+=i;
19         }
20         cout << "Case " << tt << ": " << s << endl;
21     }
22 }
23

```

## uva/10793.cpp

```

1 //10793
2 //The Orc Attack
3 //Graphs;Shortest Path;Floyd-Warshall

```

```

4  #include <iostream>
5  #include <algorithm>
6  #define MAX 105
7  using namespace std;
8
9  int G[MAX][MAX];
10
11 int main() {
12     int t; cin >> t;
13     for(int tt=1; tt<=t; tt++) {
14         int n, m; cin >> n >> m;
15         for(int i=1; i<=n; i++) {
16             for(int j=1; j<=m; j++)
17                 G[i][j] = 1<<29;
18             G[i][i] = 0;
19         }
20
21         for(int i=0; i<m; i++) {
22             int x, y, c;
23             cin >> x >> y >> c;
24             G[x][y] = G[y][x] = min(G[x][y], c);
25         }
26
27         for(int k=1; k<=n; k++)
28             for(int i=1; i<=n; i++)
29                 for(int j=1; j<=n; j++)
30                     G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
31
32         int minn = 1<<29;
33         for(int i=1; i<=n; i++)
34             if (*min_element(G[i]+1, G[i]+6) == *max_element(G[i]+1, G[i]+6))
35                 minn = min(minn, *max_element(G[i]+1, G[i]+n+1));
36
37         cout << "Map " << tt << ": ";
38         if (minn < 1<<29)
39             cout << minn << endl;
40         else
41             cout << -1 << endl;
42     }
43 }

```

## uva/10827.cpp

```

1  //10827
2  //Maximum sum on a torus
3  //Dynamic Programming;Maximum Sum Sub-rectangle
4  #include <iostream>
5  #include <climits>
6  #define MAX 160
7  using namespace std;
8
9  int T[MAX][MAX];
10
11 int main() {
12     int n, a, cases;
13     cin >> cases;
14     while(cin >> n) {
15         for(int i=1; i<=n; i++) {
16             for(int j=1; j<=n; j++) {
17                 cin >> T[i][j];
18                 T[i+n][j] = T[i][j];
19             }
20         }
21
22         for(int i=1; i<=2*n; i++)
23             for(int j=1; j<=n; j++)
24                 T[i][j] += T[i-1][j];
25
26         int t = 0;
27         for(int i=1; i<=2*n; i++) {
28             for(int j=i; j<=min(i+n-1, 2*n); j++) {
29                 int smax=0, smin=0, ssum=0, tmax=0, tmin=0;
30                 for(int k=1; k<=n; k++)
31                     ssum += T[j][k] - T[i-1][k];
32
33                 for(int k=1; k<=n; k++) {
34                     int a = T[j][k] - T[i-1][k];
35                     smax += a;
36                     smin += a;
37
38                     tmax = max(tmax, smax);
39                     tmin = min(tmin, smin);
40

```

```

41         if (smax < 0) smax = 0;
42         if (smin > 0) smin = 0;
43     }
44     t = max(t, max(tmax, ssum-tmin));
45 }
46 }
47
48     cout << t << endl;
49 }
50
51     return 0;
52 }

```

## uva/10891.cpp

```

1  //10891
2  //Game of Sum
3  //Dynamic Programming;Matrix Multiplication
4  #define MAX 101
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 maxx = INT_MIN;
18     for(int i=a; i<=b; i++)
19         maxx = max(maxx, S[b]-S[a-1] - TT(i+1,b));
20
21     for(int i=b; i>=a; i--)
22         maxx = max(maxx, S[b]-S[a-1] - TT(a,i-1));
23
24     V[a][b] = true;
25     return T[a][b] = maxx;
26 }
27
28 int main() {
29     while(cin >> n, n) {
30         memset(S, 0, sizeof(S));
31         memset(V, 0, sizeof(V));
32         S[0] = 0;
33         for(int i=1; i<=n; i++) {
34             cin >> S[i];
35             S[i] += S[i-1];
36         }
37
38         cout << 2*TT(1, n)-S[n]-S[0] << endl;
39     }
40 }
41 }

```

## uva/10930.cpp

```

1  //10930
2  //A-Sequence
3  //Dynamic Programming;Knapsack;Binary Knapsack
4  #include <iostream>
5  #include <cstring>
6  #include <iomanip>
7  using namespace std;
8
9  int K[30001];
10
11 int main() {
12     int n, t=0, w;
13     while(t++, cin >> n) {
14         cout << "Case #" << t << ":";
15         memset(K, 0, sizeof(K));
16
17         bool ok=true;
18
19         K[0] = 1; int last = 0;
20         for(int i=1; i<=n; i++) {
21             cin >> w; cout << " " << w;
22             ok &= !K[w] && w > last;

```

```

23         for(int j=10000; j>=w; j--)
24             if (K[j-w])
25                 K[j] = 1;
26         last = w;
27     }
28     cout << endl;
29     cout << "This is" << (ok?"":" not") << " an A-sequence." << endl;
30 }
31
32 return 0;
33 }

```

## uva/10986.cpp

```

1  //10986
2  //Sending email
3  //Graphs;Shortest Path;Dijkstra
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <vector>
8  #include <algorithm>
9  #include <queue>
10 #define MAX 200010
11
12 using namespace std;
13
14 struct Edge {
15     int v, c;
16     Edge(int v, int c) : v(v), c(c) {}
17     inline bool operator < (const Edge& that) const { return c > that.c; }
18 };
19
20 vector<Edge> G[MAX];
21 priority_queue<Edge> Q;
22 int n, m, s, t;
23 int V[MAX];
24
25
26 int main() {
27     int tt; cin >> tt; tt=0;
28     while(cin >> n >> m >> s >> t) {
29         int before = 0;
30         memset(V, 0x3f, sizeof(V));
31         memset(G, 0, sizeof(G));
32         Q = priority_queue<Edge>();
33
34         for(int i=0; i<m; i++) {
35             int a, b, c;
36             cin >> a >> b >> c;
37             G[a].push_back(Edge(b, c));
38             G[b].push_back(Edge(a, c));
39             before += c;
40         }
41
42         int totalc=0;
43
44         Q.push(Edge(s, 0));
45
46         while(totalc < n && !Q.empty()) {
47             Edge item = Q.top(); Q.pop();
48             if (item.c >= V[item.v]) continue;
49
50             V[item.v] = item.c;
51             totalc++;
52
53             for(int j=0; j<G[item.v].size(); j++) {
54                 Edge e = G[item.v][j];
55                 if (item.c + e.c < V[e.v])
56                     Q.push(Edge(e.v, item.c + e.c));
57             }
58         }
59
60         cout << "Case #" << ++tt << ": ";
61         if (V[t] < 0x3f3f3f3f)
62             cout << V[t] << endl;
63         else
64             cout << "unreachable" << endl;
65     }
66     return 0;
67 }

```

## uva/11003.cpp

```
1 //11003
2 //Boxes
3 //Dynamic Programming;Longest Increasing Subsequence
4 #include <iostream>
5 #include <string>
6 #include <cstring>
7 #include <cmath>
8 #include <climits>
9 #define MAX 10005
10 using namespace std;
11
12 int T[MAX];
13 int main() {
14     int n, w, c;
15     while(cin >> n, n) {
16         memset(T, 0, sizeof(T));
17
18         int k = 0;
19         T[0] = INT_MAX;
20         for(int i=1; i<=n; i++) {
21             cin >> w >> c;
22             for(int j=k; j>=0; j--) {
23                 int next = min(T[j]-w, c);
24                 if (next >= T[j+1]) {
25                     T[j+1] = next;
26                     k=max(k, j+1);
27                 }
28             }
29         }
30
31         cout << k << endl;
32     }
33
34     return 0;
35 }
```

## uva/11059.cpp

```
1 //11059
2 //Maximum Product
3 //Dynamic Programming;Maximum Sum Contiguous Subsequence
4 #include <iostream>
5 #include <climits>
6 #include <cmath>
7 #define MAX 1005
8 using namespace std;
9
10 int main() {
11     long long n, a, t=0;
12     while(cin >> n) {
13         long long maxx=0, newneg=0, newpos=0, spos=1, sneg=1;
14         bool valid = false;
15         for(int i=0; i<n; i++) {
16             cin >> a;
17             if (spos*a>0) {
18                 valid = true;
19                 spos*=a;
20             } else {
21                 newneg = spos*a;
22                 spos = 1;
23             }
24
25             if (sneg*a<0) {
26                 sneg*=a;
27             } else {
28                 if (sneg*a>0) valid = true;
29                 newpos = sneg*a;
30                 sneg = 1;
31             }
32
33             maxx = max(maxx, spos = max(spos, newpos));
34             sneg = min(sneg, newneg);
35             newpos = newneg = 0;
36         }
37         if (!valid) maxx = 0;
38         cout << "Case #" << ++t << ": The maximum product is " << maxx << "." << endl;
39         cout << endl;
40     }
41 }
```



```

42 |
43 |     return 0;
44 | }

```

## uva/11096.cpp

```

1  //11096
2  //Nails
3  //Math;Geometry;Convex Hull;Monotone Chain
4  #include <iostream>
5  #include <cmath>
6  #include <iomanip>
7  #include <algorithm>
8  #define ull long long
9  using namespace std;
10
11 struct Point {
12     ull x, y;
13
14     ull product(Point a, Point b) {
15         return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
16     }
17
18     bool right(Point a, Point b) {
19         return product(a, b) > 0;
20     }
21
22     double dist(Point b) {
23         return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y));
24     }
25
26     bool operator <(const Point& p) const {
27         if (this->x != p.x) return this->x < p.x;
28         return this->y < p.y;
29     }
30
31     bool operator ==(const Point& p) const {
32         return this->x == p.x and this->y == p.y;
33     }
34 };
35
36 int convexHull(Point* P, int n, Point* S) {
37     sort(P, P+n);
38
39     int m=0;
40     for(int i=0; i<n; i++) {
41         while(m >= 2 && !S[m-1].right(S[m-2], P[i])) m--;
42         S[m++] = P[i];
43     }
44     m--;
45
46     for(int i=n-1, k=m; i >= 0; i--) {
47         while(m >= k+2 && !S[m-1].right(S[m-2], P[i])) m--;
48         S[m++] = P[i];
49     }
50     m--;
51
52     return m;
53 }
54
55 Point P[120], S[120];
56
57 int main() {
58     int tt; cin >> tt;
59     while(tt--) {
60         int r, n;
61         cin >> r >> n;
62         for(int i=0; i<n; i++)
63             cin >> P[i].x >> P[i].y;
64
65         int s = convexHull(P, n, S);
66
67         double final = 0.0;
68         for(int i=0; i<s; i++)
69             final += S[i].dist(S[(i+1)%n]);
70
71         final = max(final, (double)r);
72
73         cout << fixed << setprecision(5) << final << endl;
74     }
75 }
76

```

## uva/11110.cpp

```
1 //11110
2 //Equidivisions
3 //Graphs;Flood Fill
4 #include <iostream>
5 #include <string>
6 #include <sstream>
7 #include <cstring>
8 #define MAX 102
9 using namespace std;
10
11 int G[MAX][MAX];
12 int n;
13
14 int fill(int x, int y, int v) {
15     if (G[x][y] != v) return 0;
16     if (x<=0 || x>n || y<=0 || y>n) return 0;
17
18     G[x][y] = -1;
19     return 1 +
20         fill(x-1, y, v) + fill(x+1, y, v) +
21         fill(x, y-1, v) + fill(x, y+1, v);
22 }
23
24 int main() {
25     while(cin >> n, n) {
26         int a, b; string s;
27         memset(G, 0, sizeof(G));
28
29         getline(cin, s);
30         for(int i=1;i<n;i++) {
31             getline(cin, s);
32             stringstream sin(s);
33             while(sin >> a >> b)
34                 G[a][b] = i;
35         }
36         bool good = true;
37         for(int i=1;i<=n;i++) {
38             for(int j=1;j<=n;j++) {
39                 if (G[i][j] >= 0)
40                     good &= fill(i,j, G[i][j]) == n;
41             }
42         }
43
44         cout << (good?"good":"wrong") << endl;
45     }
46     return 0;
47 }
48 }
```

## uva/11151.cpp

```
1 //11151
2 //Longest Palindrome
3 //Dynamic Programming;Longest Common Subsequence
4 #include <iostream>
5 #include <string>
6 #include <cstring>
7 #include <cmath>
8 #define MAX 1005
9 using namespace std;
10
11 int T[MAX][MAX];
12 string P, Q;
13
14 int main() {
15     int p, q, t;
16     cin >> t;
17     getline(cin, P);
18     while(t--) {
19         getline(cin, P);
20         Q = string(P.rbegin(), P.rend());
21         int p = P.size(), q = Q.size();
22
23         for(int i=0; i<=p; i++) { T[i][0] = 0; }
24         for(int i=0; i<=q; i++) { T[0][i] = 0; }
25
26         for(int i=1; i<=p; i++) {
27             for(int j=1; j<=q; j++) {
28                 if (P[i-1] == Q[j-1]) {
```

```

29         T[i][j] = T[i-1][j-1] + 1;
30     }
31     else {
32         T[i][j] = max(T[i-1][j], T[i][j-1]);
33     }
34 }
35 }
36
37     cout << T[p][q] << endl;
38 }
39
40     return 0;
41 }

```

## uva/11157.cpp

```

1  //11157
2  //Dynamic Frog
3  //Misc;Sort
4  #include <iostream>
5  #include <algorithm>
6  #include <vector>
7  using namespace std;
8
9  vector<int> V;
10
11 int main() {
12     int t, n, d;
13     cin >> t; t=0;
14
15     while(cin >> n >> d) {
16         char a; int b;
17         V.clear();
18         V.push_back(0);
19         V.push_back(d);
20         for(int i=0; i<n; i++) {
21             cin >> a; cin.ignore(); cin >> b;
22             V.push_back(b);
23             if (a=='B')
24                 V.push_back(b);
25         }
26         sort(V.begin(), V.end());
27
28         int maxx = 0;
29         for(int i=3; i<V.size(); i+=2)
30             maxx = max(maxx, V[i]-V[i-2]);
31
32         for(int i=2; i<V.size(); i+=2)
33             maxx = max(maxx, V[i]-V[i-2]);
34
35         cout << "Case " << ++t << ": " << maxx << endl;
36     }
37 }
38

```

## uva/11159.cpp

```

1  //11159
2  //Factors and Multiples
3  //Graphs;Bipartite Matching
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <string>
8  #define MAX 205
9  using namespace std;
10
11 int A[MAX], B[MAX], G[MAX][MAX], n;
12 bool V[MAX];
13
14 int send(int s, int t, int minn) {
15     V[s] = true;
16
17     if (s==t) return minn;
18     for(int i=0; i<n; i++) {
19         if (!V[i] && G[s][i] > 0) {
20             if (int sent = send(i, t, min(minn, G[s][i]))) {
21                 G[s][i] -= sent;
22                 G[i][s] += sent;
23                 return sent;
24             }
25         }
26     }
27 }

```

```

25     }
26 }
27 return 0;
28 }
29
30 int main() {
31     int t; cin >> t;
32
33     for(int tt=1; tt<=t; tt++) {
34         memset(G, 0, sizeof(G));
35         memset(V, 0, sizeof(V));
36
37         int a; cin >> a;
38         for(int i=1; i<=a; i++)
39             cin >> A[i];
40
41         int b; cin >> b;
42         for(int i=1; i<=b; i++)
43             cin >> B[i];
44
45         for(int i=1; i<=a; i++) {
46             G[0][i] = 1;
47             for(int j=1; j<=b; j++)
48                 if (B[j]==0 || A[i] != 0 && B[j]%A[i] == 0)
49                     G[i][a+j] = 1;
50         }
51
52         for(int i=a+1; i<=a+b; i++)
53             G[i][a+b+1] = 1;
54
55         n = a+b+1;
56
57         int total = 0;
58         while(int sent = send(0, n, INT_MAX)) {
59             total += sent;
60             memset(V, 0, sizeof(V));
61         }
62         cout << "Case " << tt << ": " << total << endl;
63     }
64
65     return 0;
66 }

```

## uva/11172.cpp

```

1 //579
2 //ClockHands
3 //Misc;Ad hoc
4 #include <iostream>
5 #include <iomanip>
6 #include <cmath>
7 using namespace std;
8
9 int main() {
10     int n;
11     cin >> n;
12     while(n--) {
13         int x, y;
14         cin >> x >> y;
15         if (x>y) {
16             cout << ">" << endl;
17         } else if (x<y) {
18             cout << "<" << endl;
19         } else {
20             cout << "=" << endl;
21         }
22     }
23 }

```

## uva/11235.cpp

```

1 //11235
2 //Frequent Values
3 //Misc;Segment Tree;Range Maximum Query
4 #include <iostream>
5 #include <cstring>
6 #include <cstdio>
7 #define MAX 600100
8 #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     int update(int v, int a, int b, int i, int x) {
50         if (a==b)
51             return V[a] = x, T[v] = a;
52         else if (i<=(a+b)/2)
53             return T[v] = maxv(T[v*2+1], update(v*2, a, (a+b)/2, i, x));
54         else
55             return T[v] = maxv(T[v*2], update(v*2+1, (a+b)/2+1, b, i, x));
56     }
57
58     int query(int i, int j) {
59         return query(1, 1, n, i, j);
60     }
61
62     int update(int i, int x) {
63         return update(1, 1, n, i, x);
64     }
65 };
66
67 Segtree T;
68 int S[MAX], E[MAX], C[MAX], N[MAX];
69
70 int main() {
71     int n, q;
72     while(cin >> n >> q, n) {
73         T.clear(n);
74
75         for(int i=1; i<=n; i++)
76             cin >> N[i];
77
78         for(int i=n; i>0; i--) {
79             if (i<n && N[i] == N[i+1])
80                 E[i] = E[i+1];
81             else
82                 E[i] = i;
83         }
84
85         for(int i=1; i<=n; i++) {
86             if (i>1 && N[i] == N[i-1])
87                 S[i] = S[i-1];
88             else
89                 S[i] = i;
90             C[i] = E[i] - S[i] + 1;
91             T.update(i, C[i]);
92         }
93     }
94 }

```

```

95     for(int i=0; i<q; i++) {
96         int a, b; cin >> a >> b;
97         if (N[a] == N[b])
98             cout << b-a+1 << endl;
99         else {
100             int case1 = E[a] - max(S[a], a) + 1;
101             int case2 = min(E[b], b) - S[b] + 1;
102             int case3 = C[T.query(E[a] + 1, S[b]-1)];
103             cout << max(case1, max(case2, case3)) << endl;
104         }
105     }
106 }
107 }

```

## uva/11280.cpp

```

1  //11280
2  //Flying to Fredericton
3  //Graphs;Shortest Path;Dijkstra
4  #include <iostream>
5  #include <cstring>
6  #include <string>
7  #include <vector>
8  #include <map>
9  #include <queue>
10 #include <algorithm>
11 #define MAX 1001
12 using namespace std;
13
14 struct Edge {
15     int v, c, s;
16     Edge(int v, int c, int s) : v(v), c(c), s(s) {}
17     inline bool operator <(const Edge& a) const {
18         return this->c > a.c;
19     }
20 };
21
22 vector<Edge> G[MAX];
23 int V[MAX][MAX];
24 map<string, int> S;
25 int n, m;
26
27 int main() {
28     int T; cin >> T;
29     for(int tt=1; tt<=T; tt++) {
30         S.clear();
31         memset(G, 0, sizeof(G));
32         memset(V, 0x7f, sizeof(V));
33
34         cin >> n;
35         for(int i=1; i<=n; i++) {
36             string s; cin >> s;
37             S[s] = i;
38         }
39
40         cin >> m;
41         for(int i=1; i<=m; i++) {
42             string s1, s2;
43             int cost;
44             cin >> s1 >> s2 >> cost;
45
46             G[S[s1]].push_back(Edge(S[s2], cost, 0));
47         }
48
49         priority_queue<Edge> Q;
50         Q.push(Edge(1, 0, 0));
51
52         while(!Q.empty()) {
53             Edge e = Q.top(); Q.pop();
54             if (V[e.v][e.s] < e.c) continue;
55
56             V[e.v][e.s] = e.c;
57
58             for(int i=0; i<G[e.v].size(); i++) {
59                 Edge a = G[e.v][i];
60                 Q.push(Edge(a.v, e.c+a.c, e.s+1));
61             }
62         }
63
64         if (tt>1) cout << endl;
65         cout << "Scenario #" << tt << endl;
66         int q; cin >> q;
67         for(int i=0; i<q; i++) {

```

```

68         int a; cin >> a;
69         int minn = 0x7f7f7f7f;
70         for(int j=0; j<=a+1; j++) {
71             minn = min(minn, V[n][j]);
72         }
73
74         if (minn < 0x7f7f7f7f)
75             cout << "Total cost of flight(s) is $" << minn << endl;
76         else
77             cout << "No satisfactory flights" << endl;
78     }
79 }
80 }
81 }

```

## uva/11294.cpp

```

1  //11294
2  //Wedding
3  //Graphs;2-SAT
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #include <vector>
8  #define MAX 1000
9  using namespace std;
10
11 vector<int> G[MAX*2], T[MAX*2];
12 int O[MAX*2], V[MAX*2], npv, n;
13 char R[MAX*2];
14
15 int neg(int x) {
16     if (x>=n) return x-n;
17     return x+n;
18 }
19
20 void set(int v, bool value) {
21     if (R[v] != 0) return;
22     R[v] = value ? 'w' : 'h';
23     R[neg(v)] = value ? 'h' : 'w';
24
25     if (value)
26         for(int i=0; i<G[v].size(); i++)
27             set(G[v][i], true);
28     else
29         for(int i=0; i<G[neg(v)].size(); i++)
30             set(G[neg(v)][i], true);
31 }
32
33 void DFS(int v){
34     V[v] = 1;
35     for(int i = 0; i < G[v].size(); i++)
36         if (!V[G[v][i]])
37             DFS(G[v][i]);
38     O[npv++] = v;
39 }
40
41 void DFSt(int v, int comp){
42     V[v] = comp;
43     for(int i = 0; i < T[v].size(); i++)
44         if (!V[T[v][i]])
45             DFSt(T[v][i], comp);
46 }
47
48 int main() {
49     int m;
50     while(cin >> n >> m, n||m) {
51         memset(G, 0, sizeof(G));
52         memset(T, 0, sizeof(T));
53         memset(R, 0, sizeof(R));
54
55         for(int i=0;i<m; i++) {
56             int a, b; char c, d;
57             cin >> a >> c >> b >> d;
58             if (c=='h') a=neg(a);
59             if (d=='h') b=neg(b);
60
61             G[neg(a)].push_back(b);
62             G[neg(b)].push_back(a);
63         }
64
65         G[neg(0)].push_back(0);
66

```

```

67     for(int i=0; i<2*n; i++)
68         for(int j=0; j<G[i].size(); j++)
69             T[G[i][j]].push_back(i);
70
71     npv = 0;
72     memset(V, 0, sizeof(V));
73     memset(O, 0, sizeof(O));
74
75     for(int i = 0; i < 2*n; i++)
76         if(!V[i]) DFS(i);
77
78     memset(V, 0, sizeof(V));
79
80     int comp = 0;
81     for(int i = 2*n-1; i >= 0; i--)
82         if(!V[O[i]])
83             DFSst(O[i], ++comp);
84
85     bool result = true;
86     for(int i=0; i<n; i++) {
87         result &= V[i] != V[neg(i)];
88     }
89
90     if (!result) {
91         cout << "bad luck" << endl;
92         continue;
93     }
94
95     for(int i=1; i<=comp; i++) {
96         for(int j=0; j<2*n; j++) {
97             if (V[j] == i)
98                 set(j, false);
99         }
100     }
101
102     for(int i=1; i<n; i++) {
103         if (i>1) cout << " ";
104         cout << i << R[i];
105     }
106     cout << endl;
107 }
108 }

```

## uva/11297.cpp

```

1  //11297
2  //Census
3  //Misc;Segment Tree;2D
4  #include <iostream>
5  #include <cstring>
6  #include <functional>
7  #define MAX 506
8  #define ull long long
9  using namespace std;
10
11 int P[MAX][MAX];
12
13 struct Point {
14     int x, y, mx;
15     Point() : x(0), y(0), mx(-1) {}
16     Point(int x, int y, int mx) : x(x), y(y), mx(mx) {}
17
18     bool operator <(const Point& other) const {
19         return mx < other.mx;
20     }
21 };
22
23 struct Segtree2d {
24     Point T[2*MAX*MAX];
25     int n, m;
26
27     void clear(int n, int m) {
28         this->n = n;
29         this->m = m;
30         build(1, 1, 1, n, m);
31     }
32
33     int c(int s1, int s2) {
34         return (s1+s2)/2;
35     }
36
37     Point build(int v, int a1, int b1, int a2, int b2) {
38         if (a1>a2 || b1>b2) return def();

```



```

39     if (a1 == a2 && b1 == b2)
40         return T[v] = Point(a1, b1, P[a1][b1]);
41
42
43     T[v] = def();
44     T[v] = maxv(T[v], build(4*v-2, a1, b1, c(a1, a2), c(b1, b2)));
45     T[v] = maxv(T[v], build(4*v-1, c(a1, a2)+1, b1, a2, c(b1, b2)));
46     T[v] = maxv(T[v], build(4*v+0, a1, c(b1, b2)+1, c(a1, a2), b2));
47     T[v] = maxv(T[v], build(4*v+1, c(a1, a2)+1, c(b1, b2)+1, a2, b2));
48     return T[v];
49 }
50
51 //virtual apenas para permitir árvore de mínimo
52 virtual Point maxv(Point a, Point b) {
53     return max(a, b);
54 }
55
56 virtual Point def() {
57     return Point(0, 0, -1);
58 }
59
60
61 Point query(int v, int a1, int b1, int a2, int b2, int x1, int y1, int x2, int y2) {
62     if (x1>a2 || y1>b2 || x2<a1 || y2<b1 || a1>a2 || b1>b2)
63         return def();
64
65     if (x1<=a1 && y1<=b1 && a2<=x2 && b2<=y2)
66         return T[v];
67
68     Point mx = def();
69
70     mx = maxv(mx, query(4*v-2, a1, b1, c(a1, a2), c(b1, b2), x1, y1, x2, y2));
71     mx = maxv(mx, query(4*v-1, c(a1, a2)+1, b1, a2, c(b1, b2), x1, y1, x2, y2));
72     mx = maxv(mx, query(4*v+0, a1, c(b1, b2)+1, c(a1, a2), b2, x1, y1, x2, y2));
73     mx = maxv(mx, query(4*v+1, c(a1, a2)+1, c(b1, b2)+1, a2, b2, x1, y1, x2, y2));
74
75     return mx;
76 }
77
78 Point query(int x1, int y1, int x2, int y2) {
79     return query(1, 1, 1, n, m, x1, y1, x2, y2);
80 }
81
82 Point update(int v, int a1, int b1, int a2, int b2, int x, int y, int value) {
83     if (a1>a2 || b1>b2) return def();
84
85     if (x>a2 || y>b2 || x<a1 || y<b1)
86         return T[v];
87
88     if (x==a1 && y==b1 && x==a2 && y==b2)
89         return T[v] = Point(x, y, value);
90
91     Point mx = def();
92
93     mx = maxv(mx, update(4*v-2, a1, b1, c(a1, a2), c(b1, b2), x, y, value));
94     mx = maxv(mx, update(4*v-1, c(a1, a2)+1, b1, a2, c(b1, b2), x, y, value));
95     mx = maxv(mx, update(4*v+0, a1, c(b1, b2)+1, c(a1, a2), b2, x, y, value));
96     mx = maxv(mx, update(4*v+1, c(a1, a2)+1, c(b1, b2)+1, a2, b2, x, y, value));
97
98     return T[v] = mx;
99 }
100
101 Point update(int x, int y, int value) {
102     return update(1, 1, 1, n, m, x, y, value);
103 }
104 };
105
106 struct Segtree2dMin : Segtree2d {
107     Point maxv(Point a, Point b) {
108         return min(a, b);
109     }
110
111     Point def() {
112         return Point(0, 0, 1<<29);
113     }
114 };
115
116 Segtree2d Tmax;
117 Segtree2dMin Tmin;
118
119 int main() {
120     int n, m;
121     while(cin >> n >> m) {
122         for(int i=1; i<=n; i++)

```

```

123         for(int j=1; j<=m; j++)
124             cin >> P[i][j];
125
126         Tmax.clear(n, m);
127         Tmin.clear(n, m);
128
129
130         int q; cin >> q;
131         while(q-->0) {
132             char cmd;
133             cin >> cmd;
134
135             if (cmd == 'q') {
136                 int x1, y1, x2, y2;
137                 cin >> x1 >> y1 >> x2 >> y2;
138                 cout << Tmax.query(x1, y1, x2, y2).mx << " " << Tmin.query(x1, y1, x2, y2).mx << endl;
139             } else {
140                 int x, y, v;
141                 cin >> x >> y >> v;
142                 Tmax.update(x, y, v);
143                 Tmin.update(x, y, v);
144             }
145         }
146     }
147 }
148 }
149 }

```

## uva/11375.cpp

```

1  //11375
2  //Matches
3  //Dynamic Programming;Ad hoc
4  #include <iostream>
5  #include <vector>
6  #include <cstring>
7  using namespace std;
8
9  int K[] = {6, 2, 5, 5, 4, 5, 6, 3, 7, 6};
10 vector<int> T[2001][10];
11
12 void add(vector<int> &a, const vector<int> &b) {
13     int carry = 0;
14     for(int i=0; i<max(a.size(), b.size()); i++) {
15         int aa = i<a.size()?a[i]:0;
16         int bb = i<b.size()?b[i]:0;
17         int cc = aa+bb+carry;
18         if (i >= a.size()) a.push_back(0);
19         a[i] = cc%10;
20         carry = cc/10;
21     }
22     if (carry)
23         a.push_back(carry);
24 }
25
26 int main() {
27     vector<int> one; one.push_back(1);
28
29     for(int i=2; i<2001; i++) {
30         for(int j=0; j<10; j++)
31             if (i>=K[j]) {
32                 add(T[i][j], one);
33                 for(int k=0; k<10; k++)
34                     add(T[i][j], T[i-K[j]][k]);
35             }
36     }
37
38     int n;
39     while(cin >> n) {
40         vector<int> ans = n>=6?one:vector<int>();
41         for(int i=1; i<10; i++)
42             add(ans, T[n][i]);
43
44         for(int i=ans.size()-1; i>=0; i--) {
45             cout << ans[i];
46         }
47         if (ans.size()==0) cout << 0;
48         cout << endl;
49     }
50
51     return 0;
52 }

```

# uva/11402.cpp

```
1 //11402
2 //Ahoy, Pirates!
3 //Misc;Segment Tree;Lazy Propagation
4 #include <iostream>
5 #include <string>
6 #include <cstring>
7 #define MAX 3000100
8 #define ull long long
9 using namespace std;
10
11 struct Node {
12     int a, b;
13     int pending;
14
15     Node() : a(0), b(0), pending(0) {}
16     Node(int a) : a(a), b(0), pending(0) { }
17     Node(int a, int b) : a(a), b(b), pending(0) { }
18
19     Node change(int n) {
20         if (n==1) {
21             b += a;
22             a = 0;
23             pending = n;
24         } else if (n==2) {
25             a += b;
26             b = 0;
27             pending = n;
28         } else if (n==3) {
29             swap(a, b);
30             pending = 3-pending;
31         }
32
33         return *this;
34     }
35
36     Node operator +(Node x) {
37         return Node(a+x.a, b+x.b);
38     }
39 };
40
41 struct Segtree {
42     Node T[MAX];
43     int n;
44
45     void clear(int n, int *P) {
46         this->n = n;
47
48         build(1, 1, n, P);
49     }
50
51     Node build(int v, int a, int b, int *P) {
52         if (a==b)
53             return T[v] = Node(1-P[a], P[a]);
54         else
55             return T[v] =
56                 build(2*v, a, (a+b)/2, P) +
57                 build(2*v+1, (a+b)/2+1, b, P);
58     }
59
60     Node update(int v, int a, int b, int i, int j, int carry, int increment) {
61         T[v].change(carry);
62
63         if (i>b || j<a)
64             return Node(0);
65
66         if (i<=a && b<=j)
67             return T[v].change(increment);
68
69         Node answer =
70             update(v*2, a, (a+b)/2, i, j, T[v].pending, increment) +
71             update(v*2+1, (a+b)/2+1, b, i, j, T[v].pending, increment);
72
73         T[v] = T[v*2] + T[v*2+1];
74
75         return answer;
76     }
77
78     Node update(int i, int j, int inc) {
79         return update(1, 1, n, i, j, 0, inc);
80     }
81 }
```

```

82     Node query(int i, int j) {
83         return update(i, j, 0);
84     }
85
86 };
87
88 Segtree T;
89 int P[MAX];
90 string s;
91
92 int main() {
93     int cases; cin >> cases;
94
95     for(int tt=1; tt<=cases; tt++) {
96         cout << "Case " << tt << ":" << endl;
97
98         int m; cin >> m;
99         int n = 0;
100        for(int i=0; i<m; i++) {
101            int t;
102            cin >> t >> s;
103            for(int j=0; j<t; j++) {
104                for(int k=0; k<s.size(); k++) {
105                    P[++n] = s[k]-'0';
106                }
107            }
108        }
109        T.clear(n, P);
110
111        int q; cin >> q;
112        int query = 0;
113        while(q--) {
114            char cmd; int a, b;
115            cin >> cmd >> a >> b;
116            a++; b++;
117            if (cmd == 'F') {
118                T.update(a, b, 1);
119            } else if (cmd == 'E') {
120                T.update(a, b, 2);
121            } else if (cmd == 'I') {
122                T.update(a, b, 3);
123            } else {
124                Node node = T.query(a, b);
125                cout << "Q" << ++query << ": " << node.b << endl;
126            }
127        }
128    }
129 }

```

## uva/11419.cpp

```

1  //11419
2  //SAM I AM
3  //Graphs;Bipartite Matching;Konig Theorem
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <string>
8  #include <cstdio>
9  #include <vector>
10 #define MAX 2005
11 using namespace std;
12
13 string VA[MAX], VB[MAX];
14 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         for(int i=0; i<p; i++) {
67             int a, b;
68             cin >> a >> b;
69             G[R(a)][C(b)] = 1;
70             G2[R(a)].push_back(C(b));
71             G2[C(b)].push_back(R(a));
72         }
73
74         n = r+c+1;
75
76         int total = 0;
77         while(int sent = send(SOURCE(), TARGET(), INT_MAX)) {
78             total += sent;
79             memset(V, 0, sizeof(V));
80         }
81
82         V[0] = V[1] = true;
83         for(int i=1; i<=r; i++) {
84             bool inflow = false;
85             for(int j=1; j<=c; j++)
86                 inflow |= G[C(j)][R(i)];
87
88             if (!V[R(i)] && !inflow)
89                 mark(R(i), true);
90         }
91         printf("%d", total);
92         for(int i=1; i<=r; i++)
93             if (!V[R(i)]) printf(" r%d", i);
94
95         for(int i=1; i<=c; i++)
96             if (V[C(i)]) printf(" c%d", i);
97
98         printf("\n");
99     }
100 }
101
102 return 0;
103 }

```

## uva/11423.cpp

```

1 //11423
2 //Cache Simulator
3 //Misc;Fenwick Tree
4 #include <iostream>
5 #include <algorithm>
6 #include <cstring>
7 #define MAX 10000100
8 using namespace std;
9

```

```

10 struct Fenwick {
11     int T[MAX];
12     int n;
13
14     Fenwick() {
15         clear(MAX);
16     }
17
18     void clear(int n) {
19         n++;
20         memset(T, 0, n*sizeof(int));
21         this->n = n;
22     }
23
24     void adjust(int k, int v) {
25         for (; k < n; k += (k&-k))
26             T[k] += v;
27     }
28
29     int rsq(int b) {
30         int sum = 0;
31         for (; b; b -= (b&-b))
32             sum += T[b];
33         return sum;
34     }
35
36     int rsq(int a, int b) {
37         return rsq(b) - rsq(a - 1);
38     }
39 };
40
41 Fenwick T;
42 int C[40], S[40], P[1<<24];
43 int caches=0, query=0;
44
45 void access(int addr) {
46     if (P[addr]) {
47         int maxCache = T.rsq(P[addr], query);
48
49         int upto = lower_bound(C, C+caches, maxCache)-C;
50
51         for(int i=0; i<upto; i++)
52             S[i]++;
53
54         T.adjust(P[addr], -1);
55     } else {
56         for(int i=0; i<caches; i++)
57             S[i]++;
58     }
59
60     T.adjust(P[addr] = ++query, 1);
61 }
62
63 int main() {
64     cin >> caches;
65     for(int i=0; i<caches; i++)
66         cin >> C[i];
67
68     string cmd;
69     while(cin >> cmd, cmd != "END") {
70         if (cmd == "ADDR") {
71             int x;
72             cin >> x;
73             access(x);
74         } else if (cmd == "RANGE") {
75             int b, y, n;
76             cin >> b >> y >> n;
77
78             for(int i=0; i<n; i++)
79                 access(b+i*y);
80         } else {
81             for(int i=0; i<caches; i++) {
82                 if (i) cout << " ";
83                 cout << S[i];
84             }
85             cout << endl;
86             memset(S, 0, sizeof S);
87         }
88     }
89
90
91     return 0;
92 }
93

```

## uva/11475.cpp

```
1 //11475
2 //Extend to Palindrome
3 //Misc;String Matching;KMP;Suffix-Prefix
4 #include <iostream>
5 #include <string>
6 #include <cstring>
7 #define MAX 100010
8 using namespace std;
9
10 int F[MAX];
11
12 void kmp_init(string& P) {
13     F[0] = 0; F[1] = 0;
14     int i = 1, j = 0;
15     while(i < P.size()) {
16         if (P[i] == P[j])
17             F[++i] = ++j;
18         else if (j == 0)
19             F[++i] = 0;
20         else
21             j = F[j];
22     }
23 }
24
25 int kmp(string& P, string& T) {
26     kmp_init(P);
27     int i = 0, j = 0;
28     int n = T.size(), m = P.size();
29
30     while(i < n) {
31         while(j < m) {
32             if (P[j] == T[i]) {
33                 i++; j++;
34             } else break;
35         }
36         if (j == 0) i++;
37         if (i == n) return j;
38         j = F[j];
39     }
40     return 0;
41 }
42
43
44 int main() {
45     string S, P, T;
46     while(cin >> S) {
47         P = string(S.rbegin(), S.rend());
48
49         string K = S.substr(0, S.size() - kmp(P, S));
50
51         cout << S + string(K.rbegin(), K.rend()) << endl;
52     }
53 }
```

## uva/11494.cpp

```
1 //11494
2 //Queen
3 //Misc;Ad hoc
4 #include <iostream>
5 #include <cstring>
6 #include <iomanip>
7 using namespace std;
8
9 int main() {
10     int x, y, a, b;
11     while(cin >> x >> y >> a >> b, x|y|a|b) {
12         if (x==a && y==b)
13             cout << 0 << endl;
14         else if (x==a || y==b || x+y == a+b || x-y==a-b)
15             cout << 1 << endl;
16         else
17             cout << 2 << endl;
18     }
19     return 0;
20 }
```

## uva/11503.cpp

```

1 //11503
2 //Virtual Friends
3 //Misc;Union-Find
4 #include <iostream>
5 #include <map>
6 #include <string>
7 #include <cstring>
8 #include <algorithm>
9 using namespace std;
10
11 int P[200001], C[200001];
12 map<string, int> M;
13
14 int parent(int v) {
15     if (P[v] != v) {
16         int p = P[v] = parent(P[v]);
17         C[v] = C[p];
18         return p;
19     } else {
20         return v;
21     }
22 }
23
24 int person(string& s) {
25     if (M.find(s) != M.end())
26         return M[s];
27     else {
28         int r = M[s] = M.size();
29         C[r] = 1; P[r] = r;
30         return r;
31     }
32 }
33
34 int main() {
35     int t; cin >> t; t=0;
36     int n;
37
38     while(cin >> n) {
39         M.clear();
40         while(n--) {
41             string p, q;
42             cin >> p >> q;
43             int a = person(p), b=person(q);
44             int pa = parent(a), pb=parent(b);
45             if (pa==pb) {
46                 cout << C[pa] << endl;
47                 continue;
48             }
49             if (pa < pb) swap(pa, pb);
50
51             P[pb] = pa;
52             cout << (C[pa]+=C[pb]) << endl;
53         }
54     }
55 }
56

```

## uva/11512.cpp

```

1 //11512
2 //GATTACA
3 //Misc;String Matching;Suffix Array;Longest Common Prefix
4 #include <iostream>
5 #include <iomanip>
6 #include <cstring>
7 #include <string>
8 #include <cmath>
9 #define MAX 10050
10 using namespace std;
11
12 int RA[MAX], tempRA[MAX];
13 int SA[MAX], tempSA[MAX];
14 int C[MAX];
15 int Phi[MAX], PLCP[MAX], LCP[MAX];
16
17 void suffix_sort(int n, int k) {
18     memset(C, 0, sizeof C);
19
20     for (int i = 0; i < n; i++)
21         C[i + k < n ? RA[i + k] : 0]++;
22
23     int sum = 0;
24     for (int i = 0; i < max(256, n); i++) {
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531

```



```

25     int t = C[i];
26     C[i] = sum;
27     sum += t;
28 }
29
30 for (int i = 0; i < n; i++)
31     tempSA[C[SA[i] + k < n ? RA[SA[i] + k] : 0]++] = SA[i];
32
33 memcpy(SA, tempSA, n*sizeof(int));
34 }
35
36 void suffix_array(string &s) {
37     int n = s.size();
38
39     for (int i = 0; i < n; i++)
40         RA[i] = s[i] - 1;
41
42     for (int i = 0; i < n; i++)
43         SA[i] = i;
44
45     for (int k = 1; k < n; k *= 2) {
46         suffix_sort(n, k);
47         suffix_sort(n, 0);
48
49         int r = tempRA[SA[0]] = 0;
50         for (int i = 1; i < n; i++) {
51             int s1 = SA[i], s2 = SA[i-1];
52             bool equal = true;
53             equal &= RA[s1] == RA[s2];
54             equal &= RA[s1+k] == RA[s2+k];
55
56             tempRA[SA[i]] = equal ? r : ++r;
57         }
58
59         memcpy(RA, tempRA, n*sizeof(int));
60     }
61 }
62
63 void lcp(string &s) {
64     int n = s.size();
65
66     Phi[SA[0]] = -1;
67     for (int i = 1; i < n; i++)
68         Phi[SA[i]] = SA[i-1];
69
70     int L = 0;
71     for (int i = 0; i < n; i++) {
72         if (Phi[i] == -1) {
73             PLCP[i] = 0;
74             continue;
75         }
76         while (s[i + L] == s[Phi[i] + L])
77             L++;
78
79         PLCP[i] = L;
80         L = max(L-1, 0);
81     }
82
83     for (int i = 1; i < n; i++)
84         LCP[i] = PLCP[SA[i]];
85 }
86
87 int main() {
88     int tt; cin >> tt;
89     while(tt--) {
90         string s; cin >> s;
91         s += "\1";
92         suffix_array(s);
93         lcp(s);
94
95         int maxx=0, start=0, count=0, last;
96         for(int i=1; i<s.size(); i++) {
97             if (LCP[i] > maxx) {
98                 maxx = LCP[i];
99                 start = i-1;
100                 count = 2;
101             } else if (LCP[i] == maxx && start+count==i) {
102                 count++;
103             }
104         }
105
106         if (maxx > 0)
107             cout << s.substr(SA[start], maxx) << " " << count << endl;
108         else

```

```

109 |         cout << "No repetitions found!" << endl;
110 |     }
111 | }

```

## uva/11518.cpp

```

1  //11518
2  //Dominos 2
3  //Graphs;Flood Fill
4  #include <iostream>
5  #include <vector>
6  #include <cstring>
7  #define MAX 10002
8  using namespace std;
9
10 vector<int> G[MAX];
11 bool V[MAX];
12 int n,m,l;
13
14 int dfs(int v) {
15     if (V[v]) return 0;
16     V[v] = true;
17     int r = 1;
18     for(int i=0;i<G[v].size(); i++)
19         r+=dfs(G[v][i]);
20     return r;
21 }
22
23 int main() {
24     int t; cin >> t;
25     while(cin >> n >> m >> l) {
26         memset(G, 0, sizeof(G));
27         memset(V, 0, sizeof(V));
28
29         for(int i=0;i<m;i++) {
30             int a, b;
31             cin >> a >> b;
32             G[a].push_back(b);
33         }
34         int sum = 0;
35         for(int i=0;i<l;i++) {
36             int a;
37             cin >> a;
38             sum+=dfs(a);
39         }
40
41         cout << sum << endl;
42     }
43     return 0;
44 }
45 }

```

## uva/11525.cpp

```

1  //11525
2  //Permutation
3  //Misc;Fenwick Tree
4  #include <iostream>
5  #include <cstring>
6  #define MAX 50100
7  using namespace std;
8
9  struct Fenwick {
10     int T[MAX];
11     int n;
12
13     Fenwick() {
14         clear(0);
15     }
16
17     void clear(int n) {
18         n++;
19         memset(T, 0, n*sizeof(int));
20         this->n = n;
21     }
22
23     void adjust(int k, int v) {
24         for (; k < n; k += (k&-k))
25             T[k] += v;
26     }
27 }

```

```

28     int rsq(int b) {
29         int sum = 0;
30         for (; b; b -= (b&b))
31             sum += T[b];
32         return sum;
33     }
34
35     int rsq(int a, int b) {
36         return rsq(b) - rsq(a - 1);
37     }
38
39     int lower_bound(int x) {
40         int first = 0, count = n;
41         while (count > 0)
42         {
43             int step = count / 2;
44             int mid = first + step;
45
46             if (rsq(mid) < x) {
47                 first = mid + 1;
48                 count -= step + 1;
49             } else {
50                 count = step;
51             }
52         }
53         return first;
54     }
55 };
56
57 Fenwick T;
58
59 int main() {
60     int tt; cin >> tt;
61     while (tt--) {
62         int n; cin >> n;
63
64         T.clear(n);
65         for (int i = 1; i <= n; i++)
66             T.adjust(i, 1);
67
68         for (int i = 0; i < n; i++) {
69             int k; cin >> k;
70
71             int x = T.lower_bound(k + 1);
72             T.adjust(x, -1);
73
74             if (i) cout << " ";
75             cout << x;
76         }
77         cout << endl;
78     }
79
80     return 0;
81 }
82

```

## uva/11532.cpp

```

1  //11532
2  //Simple Adjacency Maximization
3  //Misc;Binary Manipulation
4  #include <iostream>
5  #include <cstring>
6  #include <iomanip>
7  using namespace std;
8
9  long long T[51][51];
10
11 int main() {
12     for (int i = 1; i <= 50; i++) {
13         for (int j = 0; i + j <= 50; j++) {
14             int p = i, q = j;
15             long long n = 0L;
16             if (p % 2 != 0 && p / 2 < q) {
17                 n = 1;
18                 p--;
19             }
20
21             for (; p > 1; p -= 2) {
22                 if (q > 0)
23                     n = (n << 3) | 5L;
24                 else
25                     n = (n << 2) | 3L;
26             }
27         }
28     }
29 }

```

```

26         q--;
27     }
28
29     if (p==1) n = (n<<1) | 1L;
30     T[i][j] = n;
31 }
32 }
33
34
35 int t, p, q;
36 cin >> t;
37 while(cin >> p >> q) {
38     cout << T[p][q] << endl;
39 }
40
41 return 0;
42 }

```

## uva/11576.cpp

```

1  //11576
2  //Scrolling Sign
3  //Misc;String Matching;KMP;Suffix-Prefix
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #define MAX 100010
8  using namespace std;
9
10 int F[MAX];
11
12 void kmp_init(string& P) {
13     F[0] = 0; F[1] = 0;
14     int i = 1, j = 0;
15     while(i<P.size()) {
16         if (P[i] == P[j])
17             F[++i] = ++j;
18         else if (j == 0)
19             F[++i] = 0;
20         else
21             j = F[j];
22     }
23 }
24
25 int kmp(string& P, string& T) {
26     kmp_init(P);
27     int i = 0, j = 0;
28     int n = T.size(), m = P.size();
29
30     while(i < n) {
31         while(j < m) {
32             if (P[j] == T[i]) {
33                 i++; j++;
34             } else break;
35         }
36         if (j == 0) i++;
37         if (i==n) return j;
38         j = F[j];
39     }
40     return 0;
41 }
42
43
44 int main() {
45     int t; cin >> t; t=t0;
46     int k, w;
47     while(cin >> k >> w) {
48         int sum = 0;
49         string Q, P = "";
50         while(w--) {
51             cin >> Q;
52             sum += k-kmp(Q, P);
53             P = Q;
54         }
55         cout << sum << endl;
56     }
57 }

```

## uva/11590.cpp

```

1 | //11590

```

```

2 //Prefix Lookup
3 //Misc;String Matching;Trie
4 #include <iostream>
5 #include <cstring>
6 #define MAXS 1500010
7 #define ull unsigned long long
8 using namespace std;
9
10 struct Trie {
11     int G[MAXS][2];
12     ull S[MAXS];
13     bool E[MAXS];
14     int stateCount;
15
16     Trie() {
17         clear();
18     }
19
20     void clear() {
21         stateCount = 0;
22         clear(stateCount++);
23     }
24
25     int clear(int state) {
26         memset(G[state], -1, sizeof G[state]);
27         S[state] = 0;
28         E[state] = false;
29         return state;
30     }
31
32     void add(string &s) {
33         int state = 0;
34         for(int i=0; i<s.size()-1; i++) {
35             S[state]++;
36
37             int next = s[i] - '0';
38
39             if (G[state][next] < 0)
40                 G[state][next] = clear(stateCount++);
41
42             state = G[state][next];
43         }
44         E[state] = true;
45     }
46 };
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 {
11     int T[MAX];
12     int n;
13
14     Fenwick() {
15         clear(MAX);
16     }
17
18     void clear(int n) {
19         n++;
20         memset(T, 0, n*sizeof(int));
21         this->n = n;
22     }
23
24     void adjust(int k, int v) {
25         for (; k < n; k += (k&-k))
26             T[k] += v;
27     }
28
29     int rsq(int b) {
30         int sum = 0;
31         for (; b; b -= (b&-b))
32             sum += T[b];
33         return sum;
34     }
35
36     int rsq(int a, int b) {
37         return rsq(b) - rsq(a - 1);
38     }
39
40     int value(int b) {
41         return rsq(b, b);
42     }
43
44     int lower_bound(int x) {
45         int first = 0, count = n;
46         while (count>0)
47         {
48             int step=count/2;
49             int mid = first+step;
50
51             if (rsq(mid) < x) {

```

```

52         first = mid+1;
53         count -= step+1;
54     } else {
55         count = step;
56     }
57 }
58 return first;
59 }
60 };
61
62 Fenwick T, Q;
63 int P[MAX], W[MAX], wn=0, I[MAX], in=0, F[MAX];
64
65 int invert(int n) {
66     int r=0;
67     while(n) {
68         r *= 10;
69         r += n%10;
70         n/=10;
71     }
72     return r;
73 }
74
75 int factors(int n, int start) {
76     if (F[n]) return F[n];
77     if (not P[n]) return F[n] = 1;
78
79     for(;;start++)
80         if (n%W[start] == 0)
81             return factors(n/W[start], start)+1;
82 }
83
84
85 int main() {
86     P[1] = true;
87     for(long long i=2; i<MAX; i++) {
88         if (P[i]) continue;
89         W[wn++] = i;
90         for(long long j=i*i; j<MAX; j+=i)
91             P[j] = true;
92     }
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     return 0;
115 }
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-->0) {
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];
55             else if (signal > 0)
56                 L[lo++] = P[i];
57             else
58                 C[ce++] = P[i];
59         }
60
61         cout << m << endl;
62         cout << P[0].x << " " << P[0].y << endl;
63
64         if (lo > 0)
65             print(L, lo);
66         else
67             print(C, ce);
68
69         cout << P[m-1].x << " " << P[m-1].y << endl;
70
71         if (up > 0)
72             printInv(U, up);
73         else
74             printInv(C, ce);
75     }
76 }
77
78 }

```

## uva/11629.cpp

```

1  //11629
2  //Ballot evaluation
3  //Misc;STL map
4  #include <iostream>
5  #include <map>
6  #include <cstring>
7  using namespace std;
8
9  map<string, int> P;
10
11 int main() {
12     int n, g;
13     while(cin >> n >> g) {
14         P.clear();
15         for(int i=0; i<n; i++) {
16             string s; int a, b;
17             cin >> s >> a; cin.get(); cin >> b;

```



```

18      P[s] = a*10+b;
19  }
20
21  for(int i=1; i<=g; i++) {
22      string s = "+"; int d=0; int r;
23
24      while(s=="+") {
25          cin >> s;
26          d += P[s];
27          cin >> s;
28      }
29      cin >> r; r*=10;
30
31      bool result;
32      if ( s=="<") result = d < r;
33      if ( s=="<=") result = d <= r;
34      if ( s==">") result = d > r;
35      if ( s==">=") result = d >= r;
36      if ( s=="=") result = d == r;
37      cout << "Guess #" << i << " was " << (result?"correct":"incorrect") << "." << endl;
38  }
39
40  }
41  }

```

**uva/11631.cpp**

```

1 //11631
2 //Dark roads
3 //Graphs;Minimum Spanning Tree;Prim;Priority Queue
4 #include <iostream>
5 #include <cstring>
6 #include <climits>
7 #include <vector>
8 #include <algorithm>
9 #include <queue>
10 #define MAX 200010
11
12 using namespace std;
13
14 struct Road {
15     int v, c;
16     Road(int v, int c) : v(v), c(c) {}
17     inline bool operator < (const Road& that) const { return c > that.c; }
18 };
19
20 vector<Road> G[MAX];
21 priority_queue<Road> Q;
22 int n, m;
23 bool V[MAX];
24
25
26 int main() {
27     while(cin >> n >> m, n|m) {
28         int before = 0;
29         memset(V, 0, sizeof(V));
30         memset(G, 0, sizeof(G));
31         Q = priority_queue<Road>();
32
33         for(int i=0; i<m; i++) {
34             int a, b, c;
35             cin >> a >> b >> c;
36             G[a].push_back(Road(b, c));
37             G[b].push_back(Road(a, c));
38             before += c;
39         }
40
41         int total = 0, totalc=0;
42
43         Q.push(Road(0, 0));
44
45         while(totalc < n) {
46             Road item = Q.top(); Q.pop();
47             if (V[item.v]) continue;
48
49             V[item.v] = true;
50             total += item.c;
51             totalc++;
52
53             for(int j=0; j<G[item.v].size(); j++)
54                 if (!V[G[item.v][j].v])
55                     Q.push(G[item.v][j]);
56         }
57     }
58 }

```

```

57 |         cout << before-total << endl;
58 |     }
59 |     return 0;
60 | }
61 |

```

## uva/11658.cpp

```

1 | //11658
2 | //Best Coalitions
3 | //Dynamic Programming;Knapsack;Binary Knapsack
4 | #include <iostream>
5 | #include <cstring>
6 | #include <iomanip>
7 | using namespace std;
8 |
9 | int K[10001], W[102];
10 |
11 | int main() {
12 |     int n, x, a, b;
13 |
14 |     while(cin >> n >> x, n|x) {
15 |         memset(K, 0, sizeof(K));
16 |
17 |         for(int i=1; i<=n; i++) {
18 |             cin >> a; cin.ignore(); cin >> b;
19 |             W[i] = a*100+b;
20 |         }
21 |
22 |         K[W[x]] = 1;
23 |         for(int i=1; i<=n; i++) {
24 |             if (i==x) continue;
25 |             for(int j=10000; j>=W[i]; j--)
26 |                 if (K[j-W[i]])
27 |                     K[j] = 1;
28 |         }
29 |
30 |         int maxx = 0;
31 |         for(int i=5001; i<=10000; i++) {
32 |             if (K[i]) {
33 |                 maxx = i;
34 |                 break;
35 |             }
36 |         }
37 |
38 |         cout << fixed << setprecision(2) << (W[x]/((double)maxx)*100.0) << endl;
39 |     }
40 |
41 |     return 0;
42 | }

```

## uva/11686.cpp

```

1 | //11686
2 | //Pick up Sticks
3 | //Graphs;Topological Sorting
4 | #include <iostream>
5 | #include <cstdio>
6 | #include <vector>
7 | #include <cstring>
8 | #define MAX 1000001
9 | using namespace std;
10 |
11 | int V[MAX];
12 | int O[MAX], npv;
13 | vector<int> G[MAX];
14 | int n, m;
15 |
16 | bool DFS(int d, int v){
17 |     V[v] = 1;
18 |
19 |     for(int i=0; i<G[v].size(); i++) {
20 |         int u = G[v][i];
21 |         if (V[u] == 1) return false;
22 |         if (!V[u] && !DFS(d, u)) return false;
23 |     }
24 |     O[++npv] = v;
25 |     V[v] = 2;
26 |     return true;
27 | }
28 |

```

```

29
30 int main() {
31     int a, b;
32     while(scanf("%d%d",&n, &m), n|m) {
33         for(int i=1;i<=n;i++) G[i].clear();
34         npv = 0;
35         memset(V, 0, sizeof(V));
36         memset(O, 0, sizeof(O));
37
38         while(m--) {
39             scanf("%d%d",&a, &b);
40             G[a].push_back(b);
41         }
42
43
44         bool ok = true;
45         int d = 0;
46         for(int i = 1; i <= n; i++)
47             if (!V[i])
48                 ok &= DFS(++d, i);
49
50         if (ok)
51             for(int i = n; i > 0; i--)
52                 printf("%d\n", O[i]);
53         else
54             printf("IMPOSSIBLE\n");
55     }
56
57     return 0;
58 }

```

## uva/11703.cpp

```

1 //11703
2 //sqrt log sin
3 //Dynamic Programming;Ad hoc
4 #include <iostream>
5 #include <cmath>
6 #include <cstring>
7 #include <cassert>
8 using namespace std;
9
10 int K[1000001];
11
12 int main() {
13     K[0] = 1;
14     for(int i=1; i<1000001; i++) {
15         int a = (int)(i-sqrt(i));
16         int b = (int)log(i);
17         int c = (int)(i*pow(sin(i), 2));
18         K[i] = (K[a] + K[b] + K[c])%1000000;
19     }
20
21     int n;
22     while(cin >> n, n>-1)
23         cout << K[n] << endl;
24
25     return 0;
26 }

```

## uva/11709.cpp

```

1 //11709
2 //Trust Groups
3 //Graphs;Strongly Connected Components
4 #include <iostream>
5 #include <map>
6 #include <string>
7 #include <cstring>
8 #define MAX 1001
9 using namespace std;
10
11 map<string, int> P;
12 int person(const string& p) {
13     if (P.find(p) != P.end())
14         return P[p];
15     else
16         return P[p] = P.size();
17 }
18
19 bool V[MAX];

```

```

20 int O[MAX], npv;
21 bool G[MAX][MAX];
22 int n, m;
23
24 void DFS(int v){
25     V[v] = true;
26     for(int i = 1; i <= n; i++)
27         if (G[v][i] && !V[i])
28             DFS(i);
29     O[++npv] = v;
30 }
31
32 void DFSt(int v){
33     V[v] = true;
34     for(int i = 1; i <= n; i++)
35         if (G[i][v] && !V[i])
36             DFSt(i);
37 }
38
39
40 int main() {
41     int a, b, t; string p, q;
42     while(cin >> n >> m, n|m) {
43         memset(G, 0, sizeof(G));
44         P.clear();
45         getline(cin, p);
46
47         for(int i=0; i<n; i++) getline(cin, p);
48
49         while(m--) {
50             getline(cin, p);
51             getline(cin, q);
52             G[person(p)][person(q)] = true;
53         }
54
55         npv = 0;
56         memset(V, 0, sizeof(V));
57         memset(O, 0, sizeof(O));
58
59         for(int i = 1; i <= n; i++)
60             if(!V[i]) DFS(i);
61
62         memset(V, 0, sizeof(V));
63
64         int comp = 0;
65         for(int i = n; i > 0; i--)
66             if(!V[O[i]]) {
67                 comp++;
68                 DFSt(O[i]);
69             }
70
71         cout << comp << endl;
72     }
73
74     return 0;
75 }

```

## uva/11733.cpp

```

1 //11733
2 //Airports
3 //Graphs;Minimum Spanning Tree;Prim;Priority Queue
4 #include <iostream>
5 #include <cstring>
6 #include <climits>
7 #include <vector>
8 #include <algorithm>
9 #include <queue>
10 #define MAX 10005
11
12 using namespace std;
13
14 struct Road {
15     int v, c;
16     Road(int v, int c) : v(v), c(c) {}
17     inline bool operator < (const Road& that) const { return c > that.c; }
18 };
19
20 vector<Road> G[MAX];
21 int CStart[MAX], CCount[MAX], nc;
22 priority_queue<Road> Q;
23 int n, m, cca;
24 bool V[MAX];

```

```

25
26 int dfs(int v) {
27     V[v] = true;
28     int acum = 1;
29     for(int i=0; i<G[v].size(); i++)
30         if (!V[G[v][i].v])
31             acum += dfs(G[v][i].v);
32     return acum;
33 }
34
35 int main() {
36     int t; cin >> t; t=0;
37     while(cin >> n >> m >> cca) {
38         memset(V, 0, sizeof(V));
39         memset(G, 0, sizeof(G));
40         nc = 0;
41
42         for(int i=0; i<m; i++) {
43             int a, b, c;
44             cin >> a >> b >> c;
45             if (c<cca) {
46                 G[a].push_back(Road(b, c));
47                 G[b].push_back(Road(a, c));
48             }
49         }
50
51         for(int i=1; i<=n; i++) {
52             if (!V[i]) {
53                 CStart[nc]=i;
54                 CCount[nc]=dfs(i);
55                 nc++;
56             }
57         }
58
59         int total = nc*cca;
60
61         for(int i=0; i<nc; i++) {
62             int totalc = 0;
63             Q = priority_queue<Road>();
64             Q.push(Road(CStart[i], 0));
65             memset(V, 0, sizeof(V));
66
67             while(totalc < CCount[i]) {
68                 Road item = Q.top(); Q.pop();
69                 if (V[item.v]) continue;
70
71                 V[item.v] = true;
72                 total += item.c;
73                 totalc++;
74
75                 for(int j=0; j<G[item.v].size(); j++)
76                     if (!V[G[item.v][j].v])
77                         Q.push(G[item.v][j]);
78             }
79         }
80
81         cout << "Case #" << ++t << ": " << total << " " << nc << endl;
82     }
83     return 0;
84 }

```

## uva/11747.cpp

```

1 //11747
2 //Heavy Cycle Edges
3 //Graphs;Minimum Spanning Tree;Prim;Priority Queue
4 #include <iostream>
5 #include <cstring>
6 #include <limits>
7 #include <vector>
8 #include <algorithm>
9 #include <queue>
10 #define MAX 10005
11
12 using namespace std;
13
14 struct Road {
15     int v, c;
16     Road(int v, int c) : v(v), c(c) {}
17     inline bool operator < (const Road& that) const { return c > that.c; }
18 };
19
20 vector<Road> G[MAX];

```

```

21 int CStart[MAX], CCount[MAX], nc;
22 priority_queue<Road> Q;
23 vector<int> R;
24 int n, m;
25 bool V[MAX];
26
27 int dfs(int v) {
28     V[v] = true;
29     int acum = 1;
30     for(int i=0; i<G[v].size(); i++)
31         if (!V[G[v][i].v])
32             acum += dfs(G[v][i].v);
33     return acum;
34 }
35
36 int main() {
37     while(cin >> n >> m, n|m) {
38         memset(V, 0, sizeof(V));
39         memset(G, 0, sizeof(G));
40         nc = 0;
41         R.clear();
42
43         for(int i=0; i<m; i++) {
44             int a, b, c;
45             cin >> a >> b >> c;
46             G[a].push_back(Road(b, c));
47             G[b].push_back(Road(a, c));
48         }
49
50         for(int i=1; i<=n; i++) {
51             if (!V[i]) {
52                 CStart[nc]=i;
53                 CCount[nc]=dfs(i);
54                 nc++;
55             }
56         }
57
58         for(int i=0; i<nc; i++) {
59             int totalc=0;
60             Q.push(Road(CStart[i], 0));
61             memset(V, 0, sizeof(V));
62
63             while(totalc < CCount[i]) {
64                 Road item = Q.top(); Q.pop();
65                 if (V[item.v]) { R.push_back(item.c); continue; }
66
67                 V[item.v] = true;
68                 totalc++;
69
70                 for(int j=0; j<G[item.v].size(); j++)
71                     if (!V[G[item.v][j].v])
72                         Q.push(G[item.v][j]);
73             }
74             while(!Q.empty()) {
75                 R.push_back(Q.top().c);
76                 Q.pop();
77             }
78         }
79         sort(R.begin(), R.end());
80         if (R.size()==0) {
81             cout << "forest" << endl;
82         } else {
83             cout << R[0];
84             for(int i=1; i<R.size(); i++)
85                 cout << " " << R[i];
86             cout << endl;
87         }
88     }
89     return 0;
90 }

```

## uva/11762.cpp

```

1 //11762
2 //Race to 1
3 //Math;Probability
4 #include <iostream>
5 #include <iomanip>
6 #define MAX 1000010
7 using namespace std;
8
9 int P[MAX], K[MAX], D[MAX][30];
10 double R[MAX];

```

```

11 int main() {
12     for(int i=2; i<MAX; i++) {
13         K[i] = K[i-1];
14
15         if (P[i] > 0) continue;
16         K[i]++;
17
18         for(int j=i, k=1; j<MAX; j+=i, k++)
19             D[j][P[j]++] = k;
20     }
21
22     R[1] = 0;
23     for(int i=2; i<MAX; i++) {
24         for(int j=0; j<P[i]; j++)
25             R[i] += R[D[i][j]];
26
27         R[i] /= P[i];
28         R[i] += (double)K[i] / P[i];
29     }
30
31     int t; cin >> t;
32     for(int tt = 1; tt<=t; tt++) {
33         int a; cin >> a;
34         cout << "Case " << tt << ": " << fixed << setprecision(10) << R[a] << endl;
35     }
36 }
37

```

## uva/11770.cpp

```

1  //11770
2  //Lighting Away
3  //Graphs;Topological Sorting
4  #include <iostream>
5  #include <vector>
6  #include <cstring>
7  #define MAX 10001
8  using namespace std;
9
10 bool V[MAX];
11 int O[MAX], npv;
12 vector<int> G[MAX];
13 int n, m;
14
15 void DFS(int v){
16     if (V[v]) return;
17     V[v] = true;
18     for(int i = 0; i < G[v].size(); i++)
19         DFS(G[v][i]);
20     O[++npv] = v;
21 }
22
23 void DFSt(int v){
24     if (V[v]) return;
25     V[v] = true;
26     for(int i = 0; i < G[v].size(); i++)
27         DFSt(G[v][i]);
28 }
29
30
31 int main() {
32     int a, b;
33     int t; cin >> t; t=0;
34     while(cin >> n >> m) {
35         memset(G, 0, sizeof(G));
36
37         while(m--) {
38             cin >> a >> b;
39             G[a].push_back(b);
40         }
41
42         npv = 0;
43         memset(V, 0, sizeof(V));
44         memset(O, 0, sizeof(O));
45
46         for(int i = 1; i <= n; i++)
47             if(!V[i]) DFS(i);
48
49         memset(V, 0, sizeof(V));
50
51         int comp = 0;
52         for(int i = n; i > 0; i--)
53             if(!V[O[i]]) {

```

```

54         comp++;
55         DFSt(0[i]);
56     }
57
58     cout << "Case " << ++t << ": " << comp << endl;
59 }
60
61 return 0;
62 }

```

## uva/11833.cpp

```

1  //11833
2  //Route Change
3  //Graphs;Shortest Path;Dijkstra
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <vector>
8  #include <algorithm>
9  #include <queue>
10 #define MAX 252
11
12 using namespace std;
13
14 struct Edge {
15     int v, c;
16     Edge(int v, int c) : v(v), c(c) {}
17     inline bool operator < (const Edge& that) const { return c > that.c; }
18 };
19
20 int G[MAX][MAX];
21 int V[MAX], S[MAX];
22 int n, m, cc, kk;
23
24 int main() {
25     while(cin >> n >> m >> cc >> kk, n|m|cc|kk) {
26         memset(V, 0x3f, sizeof(V));
27         memset(S, 0, sizeof(S));
28         memset(G, -1, sizeof(G));
29
30         for(int i=0; i<m; i++) {
31             int a, b, c;
32             cin >> a >> b >> c;
33             G[a][b] = G[b][a] = c;
34         }
35
36         for(int i=cc-2; i>=0; i--) {
37             S[i] = S[i+1] + G[i][i+1];
38         }
39
40         int totalc=0;
41
42         priority_queue<Edge> Q;
43         Q.push(Edge(kk, 0));
44
45         while(totalc < n && !Q.empty()) {
46             Edge item = Q.top(); Q.pop();
47             if (item.c >= V[item.v]) continue;
48             V[item.v] = item.c;
49             totalc++;
50             if (item.v < cc) continue;
51             for(int j=0; j<n; j++) {
52                 if (G[item.v][j]>=0) {
53                     Edge e = Edge(j, G[item.v][j]);
54                     if (item.c + e.c < V[e.v])
55                         Q.push(Edge(e.v, item.c + e.c));
56                 }
57             }
58         }
59
60         int minn = 0x3f3f3f3f;
61         for(int i=0; i<cc; i++) {
62             minn = min(minn, V[i]+S[i]);
63         }
64         cout << minn << endl;
65     }
66     return 0;
67 }

```

## uva/11838.cpp



```

1 //11838
2 //Come and Go
3 //Graphs;Strongly Connected Components
4 #include <iostream>
5 #include <cstring>
6 #define MAX 1001
7 using namespace std;
8
9 bool V[MAX];
10 int O[MAX], npv;
11 bool G[MAX][MAX];
12 int n, m;
13
14 void DFS(int v){
15     V[v] = true;
16     for(int i = 1; i <= n; i++)
17         if (G[v][i] && !V[i])
18             DFS(i);
19     O[++npv] = v;
20 }
21
22 void DFSt(int v){
23     V[v] = true;
24     for(int i = 1; i <= n; i++)
25         if (G[i][v] && !V[i])
26             DFSt(i);
27 }
28
29
30 int main() {
31     int a, b, t;
32     while(cin >> n >> m, n|m) {
33         memset(G, 0, sizeof(G));
34
35         while(m--) {
36             cin >> a >> b >> t;
37             G[a][b] = true;
38             if (t==2)
39                 G[b][a] = true;
40         }
41
42         npv = 0;
43         memset(V, 0, sizeof(V));
44         memset(O, 0, sizeof(O));
45
46         for(int i = 1; i <= n; i++)
47             if(!V[i]) DFS(i);
48
49         memset(V, 0, sizeof(V));
50
51         int comp = 0;
52         for(int i = n; i > 0; i--)
53             if(!V[O[i]]) {
54                 comp++;
55                 DFSt(O[i]);
56             }
57
58         cout << (comp==1) << endl;
59     }
60
61     return 0;
62 }

```

## uva/11857.cpp

```

1 //11857
2 //Driving Range
3 //Graphs;Minimum Spanning Tree;Kruskal
4 #include <iostream>
5 #include <cstring>
6 #include <vector>
7 #include <algorithm>
8 #include <cassert>
9 using namespace std;
10
11 struct Edge {
12     int x, y, v;
13     inline bool operator <(const Edge& that) const {
14         return this->v < that.v;
15     }
16 };
17
18 Edge E[1000005];

```

```

19 int P[1000005];
20
21 inline int findset(int v) {
22     if (P[v] != v)
23         return P[v] = findset(P[v]);
24     return v;
25 }
26
27 inline int unionset(int x, int y) {
28     int a = findset(x), b = findset(y);
29     if (a==b) return -1;
30     if (a>b) swap(a,b);
31     P[b] = a;
32     return a;
33 }
34
35 int main() {
36     int n, m;
37     while(cin >> n >> m, n||m) {
38         for(int i=0; i<n; i++)
39             P[i] = i;
40
41         for(int i=0; i<m; i++)
42             cin >> E[i].x >> E[i].y >> E[i].v;
43
44         sort(E, E+m);
45
46         int maxx=0, count=0;
47         for(int i=0; i<m && count < n-1; i++) {
48             if(unionset(E[i].x, E[i].y) != -1) {
49                 maxx = max(maxx, E[i].v);
50                 count++;
51             }
52         }
53         if (count == n-1)
54             cout << maxx << endl;
55         else
56             cout << "IMPOSSIBLE" << endl;
57     }
58 }

```

## uva/11966.cpp

```

1  //11966
2  //Galactic Bonding
3  //Misc;Union-Find
4  #include <iostream>
5  #include <map>
6  #include <string>
7  #include <cstring>
8  #include <algorithm>
9  #include <cmath>
10 using namespace std;
11
12 int P[1000];
13 double X[1000], Y[1000];
14
15 inline int findset(int v) {
16     if (P[v] == v) return v;
17     return P[v] = findset(P[v]);
18 }
19
20 inline bool unionset(int x, int y) {
21     int a = findset(x), b = findset(y);
22     if (a==b) return false;
23     P[b] = a;
24     return true;
25 }
26
27 inline double dist(int a, int b) {
28     return pow(X[a]-X[b], 2.0)+pow(Y[a]-Y[b], 2.0);
29 }
30
31 int main() {
32     int t; cin >> t; t=0;
33
34     int n; double d;
35     while(cin >> n >> d) {
36         for(int i=0; i<n; i++) P[i] = i;
37
38         int sets = n;
39         for(int i=0; i<n; i++) {
40             cin >> X[i] >> Y[i];

```

```

41         for(int j=0;j<i;j++)
42             if (dist(i,j)<=d*d && unionset(i, j))
43                 sets--;
44     }
45
46     cout << "Case " << ++t << ": " << sets << endl;
47 }
48 }

```

## uva/12086.cpp

```

1  //12086
2  //Potentiometers
3  //Misc;Fenwick Tree
4  #include <iostream>
5  #include <cstring>
6  #include <string>
7  #define MAX 200100
8  using namespace std;
9
10 struct Fenwick {
11     int 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(int));
21         this->n = n;
22     }
23
24     void adjust(int k, int v) {
25         for (; k < n; k += (k&-k))
26             T[k] += v;
27     }
28
29     void update(int k, int v) {
30         adjust(k, v-rsq(k, k));
31     }
32
33     int rsq(int b) {
34         int sum = 0;
35         for (; b; b -= (b&-b))
36             sum += T[b];
37         return sum;
38     }
39
40     int rsq(int a, int b) {
41         return rsq(b) - rsq(a - 1);
42     }
43 };
44
45 Fenwick T;
46
47 int main() {
48     int n, tt=0;
49     while(cin >> n, n) {
50         if (tt++) cout << endl;
51         cout << "Case " << tt << ":" << endl;
52
53         T.clear(n);
54         for(int i=1; i<=n; i++) {
55             int a; cin >> a;
56             T.adjust(i, a);
57         }
58
59         string cmd;
60         while(cin >> cmd, cmd!="END") {
61             int a, b; cin >> a >> b;
62             if (cmd == "S") {
63                 T.update(a, b);
64             } else {
65                 cout << T.rsq(a, b) << endl;
66             }
67         }
68     }
69 }
70 }

```

## uva/12101.cpp

```
1 //12101
2 //Prime Path
3 //Graphs;Shortest Path;BFS
4 #include <iostream>
5 #include <queue>
6 #include <cstring>
7 #include <string>
8 using namespace std;
9
10 bool P[10000], V[10000];
11
12 struct Step {
13     int a, b, c, d, w;
14     Step() {}
15     Step(int a, int b, int c, int d, int w) : a(a), b(b), c(c), d(d), w(w) {}
16
17     int number() { return a*1000+b*100+c*10+d; }
18     bool valid() { return a && P[number()]; }
19
20     Step atA(int n) { return Step(n, b, c, d, w+1); }
21     Step atB(int n) { return Step(a, n, c, d, w+1); }
22     Step atC(int n) { return Step(a, b, n, d, w+1); }
23     Step atD(int n) { return Step(a, b, c, n, w+1); }
24 };
25
26 Step makestep(int n) {
27     int a, b, c, d;
28     d = n%10; n/=10;
29     c = n%10; n/=10;
30     b = n%10; n/=10;
31     a = n%10; n/=10;
32     return Step(a,b,c,d,0);
33 }
34
35 int main() {
36     memset(P, true, sizeof(P));
37     P[0] = P[1] = false;
38     for(int i=2; i<10000; i++) {
39         if (P[i]) {
40             for(int j=i*i; j<10000; j+=i)
41                 P[j] = false;
42         }
43     }
44     int t, a, b;
45     cin >> t;
46     while(cin >> a >> b) {
47         memset(V, 0, sizeof(V));
48         queue<Step> Q;
49         Q.push(makestep(a));
50         bool found = false;
51         while(!Q.empty()) {
52             Step step = Q.front(); Q.pop();
53             int n = step.number();
54             if (V[n]) continue;
55             V[n] = true;
56             if (n == b) {
57                 cout << step.w << endl;
58                 found = true;
59                 break;
60             }
61             for(int i=0; i<=9; i++) {
62                 Step sa = step.atA(i);
63                 Step sb = step.atB(i);
64                 Step sc = step.atC(i);
65                 Step sd = step.atD(i);
66                 if (sa.valid()) Q.push(sa);
67                 if (sb.valid()) Q.push(sb);
68                 if (sc.valid()) Q.push(sc);
69                 if (sd.valid()) Q.push(sd);
70             }
71         }
72         if (!found) cout << "Impossible" << endl;
73     }
74 }
75 }
```

## uva/12103.cpp

```
1 //12103
```

```

2 //Leonardo's Notebook
3 //Misc;Permutation Cycle
4 #include <iostream>
5 #include <string>
6 #include <cstring>
7 using namespace std;
8
9 int main() {
10     int t; cin >> t;
11     while(t--) {
12         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         }
62         cout << endl;
63     }
64 }
65 }

```

## uva/12137.cpp

```

1  //12137
2  //Puzzles of Triangles
3  //Math;Prime Factorization
4  #include <string.h>
5  #include <stdio.h>
6  #define PP 20000
7  #define ull unsigned long long
8
9  int W[PP], wn=0;
10 bool P[PP];
11
12 inline ull div(const ull& a, const ull& b, ull &r) {
13     r = a/b;
14     return a-r*b;
15 }
16
17 inline ull pow(const ull& a, const int b) {
18     if (b==0) return 1;
19     ull tmp = b&1 ? a : 1;
20     ull r = pow(a, b>>1);
21     return tmp*r*r;
22 }
23
24 int main() {
25     for(long long i=2; i*i<PP; i++) {
26         if (P[i]) continue;
27         W[wn++] = i;
28         for(long long j=i*i; j<PP; j+=i) {
29             P[j] = true;
30         }
31     }
32
33     ull n;
34     int t=0;
35     while(scanf("%llu", &n), n) {
36         ull ncopy = n;
37         ull step = 1;
38         for(int i=0; ncopy>1 && i<wn; i++) {
39             int power=0;
40             ull divr;
41             while(div(ncopy, W[i], divr)==0) {
42                 ncopy = divr;
43                 power++;
44             }
45             step *= pow(W[i], (power+1)/2);
46         }
47         step *= ncopy;
48
49         ull result;
50         if (div(n, step, result)==0) result--;
51         result *= 8;
52
53         if(result)
54             printf("Case %d: %llu\n", ++t, result);
55         else
56             printf("Case %d: Impossible\n", ++t);
57     }
58 }

```

## uva/12144.cpp

```

1  //12144
2  //Almost Shortest Path
3  //Graphs;Shortest Path;Dijkstra
4  #include <iostream>

```

```

5  #include <cstring>
6  #include <climits>
7  #include <vector>
8  #include <algorithm>
9  #include <queue>
10 #define MAX 501
11
12 using namespace std;
13
14 struct Edge {
15     int u, v, c;
16     Edge(int u, int v, int c) : u(u), v(v), c(c) {}
17     inline bool operator < (const Edge& that) const { return c > that.c; }
18 };
19
20 int G[MAX][MAX];
21 int V[MAX];
22 vector<int> D[MAX];
23 int n, m, s, t;
24
25 void remove(int t) {
26     if (D[t].size() == 0 || t == D[t][0]) return;
27     for(int i=0; i<D[t].size(); i++) {
28         G[D[t][i]][t] = 0;
29         remove(D[t][i]);
30     }
31 }
32
33 int shortest() {
34     memset(V, 0x3f, sizeof(V));
35     memset(D, 0, sizeof(D));
36     priority_queue<Edge> Q;
37     Q.push(Edge(s, s, 0));
38
39     while(!Q.empty()) {
40         Edge item = Q.top(); Q.pop();
41         if (item.c > V[item.v]) continue;
42         V[item.v] = item.c;
43         D[item.v].push_back(item.u);
44
45         for(int j=0; j<n; j++) {
46             if (G[item.v][j]) {
47                 Edge e = Edge(item.v, j, item.c+G[item.v][j]);
48                 if (e.c <= V[e.v])
49                     Q.push(e);
50             }
51         }
52     }
53     remove(t);
54     if (V[t] < 0x3f3f3f3f)
55         return V[t];
56     else
57         return -1;
58 }
59
60
61 int main() {
62     while(cin >> n >> m, n|m) {
63         cin >> s >> t;
64         memset(G, 0, sizeof(G));
65
66         for(int i=0; i<m; i++) {
67             int a, b, c;
68             cin >> a >> b >> c;
69             G[a][b] = c;
70         }
71
72         shortest();
73         cout << shortest() << endl;
74     }
75     return 0;
76 }

```

## uva/12147.cpp

```

1  //12147
2  //DNA Sequences
3  //Dynamic Programming;Longest Common Subsequence
4  #include <iostream>
5  #include <string>
6  #include <cstring>
7  #include <cmath>
8  #define MAX 1005

```

```

9   using namespace std;
10
11  int T[MAX][MAX];
12  int S[MAX][MAX];
13  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

```
1 //12155
2 //ASCII Diamondi
3 //Misc;Ad hoc
4 #include <iostream>
5 using namespace std;
6
7 inline int abs(int n) { return n>0?n:-n; }
8
9 inline char charAt(int n, int x, int y) {
10     x%=n*2-1; y%=n*2-1;
11     int dist = abs(n-x-1)+abs(n-y-1);
12     if (dist < n)
13         return (char)(dist%26+'a');
14     else
15         return '.';
16 }
17
18 int main() {
19     int n, ax, ay, bx, by, t=0;
20     while(cin >> n, n) {
21         cin >> ax >> ay >> bx >> by;
22         cout << "Case " << ++t << ":" << endl;
23         for(int i=ax; i<=bx; i++) {
24             for(int j=ay; j<=by; j++) {
25                 cout << charAt(n, i, j);
26             }
27             cout << endl;
28         }
29     }
30 }
31 }
```

## uva/12159.cpp

```
1 //12159
2 //Gun Fight
3 //Graphs;Bipartite Matching
4 #include <iostream>
5 #include <iomanip>
6 #include <cstring>
7 #include <vector>
8 #include <cmath>
9 #include <climits>
10 #include <vector>
11 #include <cassert>
12 #define MAX 306
13 using namespace std;
14
15 int X[MAX], Y[MAX], P[MAX], G[MAX][MAX], n, r, a, b;
16 bool V[MAX];
17
18 bool team(int c) {
19     return (X[b] - X[a])*(Y[c] - Y[a]) - (Y[b] - Y[a])*(X[c] - X[a]) > 0;
20 }
21
22 int sqrdist(int a, int b) {
23     return (X[a]-X[b])*(X[a]-X[b])+(Y[a]-Y[b])*(Y[a]-Y[b]);
24 }
25
26 int send(int s, int t, int minn) {
27     V[s] = true;
28
29     if (s==t) return minn;
30     for(int i=0; i<n; i++) {
31         if (!V[i] && G[s][i] > 0) {
32             if (int sent = send(i, t, min(minn, G[s][i]))) {
33                 G[s][i] -= sent;
34                 G[i][s] += sent;
35                 return sent;
36             }
37         }
38     }
39     return 0;
40 }
41
42 int main() {
43     int t=0;
44     while(cin >> n, n) {
45         memset(G, 0, sizeof(G));
```

```

46     for(int i=1;i<=n;i++)
47         cin >> X[i] >> Y[i] >> P[i];
48     cin >> a >> b >> r;
49
50     vector<int> A, B;
51     for(int i=1;i<=n;i++) {
52         if (P[i] == 0) continue;
53         if (team(i))
54             B.push_back(i);
55         else
56             A.push_back(i);
57     }
58     if (A.size() > B.size()) A.swap(B);
59
60     for(int i=0; i<A.size(); i++) {
61         int u=A[i];
62         G[0][u] = 1;
63         for(int j=0; j<B.size(); j++) {
64             int v = B[j];
65             G[v][n+1] = 1;
66             if (sqrdist(u, v) <= r*r && P[u] > P[v])
67                 G[u][v] = 1;
68         }
69     }
70     n++;
71
72     memset(V, 0, sizeof(V));
73     int total = 0;
74     while(int sent = send(0, n, INT_MAX)) {
75         total += sent;
76         memset(V, 0, sizeof(V));
77     }
78     cout << "Case " << ++t << ": " << total << endl;
79 }
80 }
81

```

## uva/12160.cpp

```

1  //12160
2  //Unlock the Lock
3  //Graphs;Shortest Path;BFS
4  #include <iostream>
5  #include <queue>
6  #include <cstring>
7  #include <string>
8  using namespace std;
9
10 bool V[10000];
11 int R[10];
12
13 struct Step {
14     int x, w;
15     Step() {}
16     Step(int x, int w) : x(x), w(w) {}
17
18     Step sum(int n) {
19         return Step((x+n)%10000, w+1);
20     }
21 };
22
23 int main() {
24     int a, b, n, t=0;
25     while(cin >> a >> b >> n, a|b|n) {
26         for(int i=0;i<n;i++)
27             cin >> R[i];
28
29         cout << "Case " << ++t << ": ";
30         memset(V, 0, sizeof(V));
31         queue<Step> Q;
32         Q.push(Step(a, 0));
33         bool found = false;
34         while(!Q.empty()) {
35             Step step = Q.front(); Q.pop();
36             if (V[step.x]) continue;
37             V[step.x] = true;
38             if (step.x == b) {
39                 cout << step.w << endl;
40                 found = true;
41                 break;
42             }
43             for(int i=0;i<n;i++)
44                 Q.push(step.sum(R[i]));
45         }
46     }
47 }

```

```

45     }
46     if (!found) cout << "Permanently Locked" << endl;
47 }
48
49 }

```

## uva/12168.cpp

```

1  //12168
2  //Cat vs. Dog
3  //Graphs;Bipartite Matching;Konig Theorem
4  #include <iostream>
5  #include <cstring>
6  #include <climits>
7  #include <string>
8  #define MAX 505
9  using namespace std;
10
11 string V1[MAX], V2[MAX];
12 int G[MAX][MAX], n;
13 bool V[MAX];
14
15 int send(int s, int t, int minn) {
16     V[s] = true;
17
18     if (s==t) return minn;
19     for(int i=0; i<=n; i++) {
20         if (!V[i] && G[s][i] > 0) {
21             if (int sent = send(i, t, min(minn, G[s][i]))) {
22                 G[s][i] -= sent;
23                 G[i][s] += sent;
24                 return sent;
25             }
26         }
27     }
28     return 0;
29 }
30
31 int main() {
32     int t; cin >> t;
33     int c, d, v;
34
35     while(cin >> c >> d >> v, t--) {
36         memset(G, 0, sizeof(G));
37         memset(V, 0, sizeof(V));
38
39         string s1, s2;
40         for(int i=1; i<=v; i++) {
41             cin >> s1 >> s2;
42             V1[i] = s1; V2[i] = s2;
43
44             bool dog = s1[0] == 'D';
45
46             if (dog)
47                 G[0][i] = 1;
48             else
49                 G[i][v+1] = 1;
50
51             for(int j=1; j<i; j++) {
52                 if (s1 == V2[j] || s2 == V1[j])
53                     if (dog)
54                         G[i][j] = 1;
55                     else
56                         G[j][i] = 1;
57             }
58         }
59         n = v+1;
60
61         int total = 0;
62         while(int sent = send(0, n, INT_MAX)) {
63             total += sent;
64             memset(V, 0, sizeof(V));
65         }
66         cout << v-total << endl;
67     }
68
69     return 0;
70 }

```

## uva/12172.cpp

```

1 //12172
2 //Matchsticks
3 //Misc;Greed
4 #include <iostream>
5 #include <cstring>
6 #include <cmath>
7 #define MAX 101
8 using namespace std;
9
10 void printMax(int n) {
11     if (n&1) { cout << "7"; n-=3; }
12     for(;n;n-=2) cout << "1";
13 }
14
15 void printMin(int n) {
16     switch(n) {
17         case 2: cout << "1"; return;
18         case 3: cout << "7"; return;
19         case 4: cout << "4"; return;
20         case 5: cout << "2"; return;
21         case 6: cout << "6"; return;
22     }
23
24     switch(n%7) {
25         case 1: cout << "10"; n-=8; break;
26         case 2: cout << "1"; n-=2; break;
27         case 3:
28             if (n==10) {
29                 cout << "22"; n-= 10;
30             } else{
31                 cout << "200"; n-=17;
32             }
33             break;
34         case 4: cout << "20"; n-= 11; break;
35         case 5: cout << "2"; n-= 5; break;
36         case 6: cout << "6"; n-= 6; break;
37     }
38     for(;n;n-=7) cout << "8";
39 }
40
41 int main() {
42
43
44     int n;
45     int t; cin >> t; t=0;
46
47     while(cin >> n) {
48         printMin(n);
49         cout << " ";
50         printMax(n);
51         cout << endl;
52     }
53
54     return 0;
55 }

```

## uva/12179.cpp

```

1 //12179
2 //Randomly-priced Tickets
3 //Graphs;Shortest Path;Floyd-Warshall
4 #include <iostream>
5 #include <cstring>
6 #include <cmath>
7 #include <iomanip>
8 #define MAX 101
9 using namespace std;
10
11 int G[MAX][MAX], n, r, c;
12 double P[101][10001];
13
14 int main() {
15     int t; cin >> t; t=0;
16     cout << fixed << setprecision(6);
17
18     while(cin >> n >> r) {
19         memset(G, 0x3F, sizeof(G));
20         memset(P, 0, sizeof(P));
21
22         char cc;
23         for(int i=0; i<n; i++) {
24             for(int j=0; j<n; j++) {
25                 cin >> cc;

```

```

26         if (cc=='Y') G[i][j] = 1;
27     }
28 }
29
30 for(int k=0; k<n; k++)
31     for(int i=0; i<n; i++)
32         for(int j=0; j<n; j++)
33             G[i][j] = min(G[i][j], G[i][k] + G[k][j]);
34
35 P[0][0] = 1;
36 double pp = 1.0/r;
37 for(int i=1; i<=100; i++)
38     for(int k=1; k<=r; k++)
39         for(int j=k; j<=100*r; j++)
40             P[i][j] += P[i-1][j-k] * pp;
41
42 cout << "Case " << ++t << endl;
43 cin >> c;
44 while(c--) {
45     int a, b, m;
46     cin >> a >> b >> m;
47     a--; b--;
48
49     int d=G[a][b];
50
51     double total = 0;
52     for(int i=0; i<=m; i++)
53         total += P[d][i];
54     cout << total << endl;
55 }
56 cout << endl;
57 }
58 return 0;
59 }

```

## uva/12184.cpp

```

1  //12184
2  //Transcribed Books
3  //Math;GCD
4  #include <iostream>
5  using namespace std;
6
7  long gcd(long a, long b) {
8      while(b) {
9          long c = a%b;
10         a = b;
11         b = c;
12     }
13     return a;
14 }
15
16 int main() {
17     int t; cin >> t;
18     int n;
19     while(cin >> n) {
20         long result = 0;
21         long maxSerial = 0;
22         for(int i=0; i<n; i++) {
23             long s=0, tmp;
24             for(int j=0; j<9; j++) {
25                 cin >> tmp; s+=tmp;
26             }
27             cin >> tmp;
28             s -= tmp;
29             maxSerial = max(maxSerial, tmp);
30             result = gcd(result, s);
31         }
32         if (result>1 && maxSerial < result)
33             cout << result << endl;
34         else
35             cout << "impossible" << endl;
36     }
37 }

```

## uva/12186.cpp

```

1  //12186
2  //Another Crisis
3  //Graphs;DFS
4  #include <iostream>

```

```

5  #include <vector>
6  #include <algorithm>
7  #include <cstring>
8  #include <cmath>
9  #define MAX 100002
10 using namespace std;
11
12 vector<int> G[MAX];
13 int n, t;
14
15 int dfs(int v) {
16     if (G[v].empty()) return 1;
17     vector<int> mins;
18     for(int i=0; i<G[v].size(); i++)
19         mins.push_back(dfs(G[v][i]));
20     sort(mins.begin(), mins.end());
21
22     int get = (int)ceil(G[v].size()*t/100.0);
23     int sum = 0;
24     for(int i=0; i<get; i++) sum+=mins[i];
25     return sum;
26 }
27
28 int main() {
29     int boss;
30     while(cin >> n >> t, n|t) {
31         memset(G, 0, sizeof(G));
32         for(int i=1; i<=n; i++) {
33             cin >> boss; G[boss].push_back(i);
34         }
35         cout << dfs(0) << endl;
36     }
37     return 0;
38 }

```

## uva/12189.cpp

```

1  //12189
2  //Dinner Hall
3  //Misc;Sort
4  #include <iostream>
5  #include <vector>
6  #include <algorithm>
7  using namespace std;
8
9  struct Event {
10     int s; char t;
11     Event() {}
12     Event(int s, char t) : s(s), t(t) {}
13     int entry() { return t=='E'?1:0; }
14     int exit() { return t=='X'?1:0; }
15     int unknown() { return t=='?'?1:0; }
16 };
17
18 bool compare(const Event& a, const Event& b) {
19     return a.s < b.s;
20 }
21
22 vector<Event> V;
23
24 int main() {
25     int n;
26     while(cin >> n, n) {
27         int entries=0, exits=0, unknowns=0;
28         int a, b, c; char t;
29         V.clear();
30         for(int i=0; i<n; i++) {
31             cin >> a >> t >> b >> t >> c >> t;
32             Event e = Event(a*60+b*60+c, t);
33             entries += e.entry();
34             exits += e.exit();
35             unknowns += e.unknown();
36             V.push_back(e);
37         }
38         sort(V.begin(), V.end(), compare);
39
40         int maxEntries = (unknowns-(entries-exits))/2;
41         int maxx = 0, current=0;
42         for(int i=0; i<V.size(); i++) {
43             if (V[i].entry()) current++;
44             if (V[i].exit()) current--;
45             if (V[i].unknown()) {
46                 if (maxEntries) { current++; maxEntries--; }

```

```

47         else { current--; }
48     }
49     maxx = max(maxx, current);
50 }
51 cout << maxx << endl;
52 }
53 }

```

## uva/12190.cpp

```

1  //12190
2  //Electric Bill
3  //Misc;Binary Search
4  #include <iostream>
5  using namespace std;
6
7  int C(int price) {
8      int cons = 0;
9      cons += min(max(0, price/2), 100); price -= 2*100;
10     cons += min(max(0, price/3), 9900); price -= 3*9900;
11     cons += min(max(0, price/5), 990000); price -= 5*990000;
12     cons += max(0, price/7);
13     return cons;
14 }
15
16 int V(int cons) {
17     int price = 0;
18     price += min(max(0, cons*2), 2*100); cons -= 100;
19     price += min(max(0, cons*3), 3*9900); cons -= 9900;
20     price += min(max(0, cons*5), 5*990000); cons -= 990000;
21     price += max(0, cons*7);
22     return price;
23 }
24
25 int main() {
26     int a, b;
27     while(cin >> a >> b, a|b) {
28         int total = C(a);
29         int begin = 0, end = total;
30         int answer = 0;
31         while(begin < end) {
32             int mine = (begin+end)/2;
33             int diff = V(total-mine)-V(mine);
34             if (diff > b)
35                 begin = mine;
36             else if (diff < b)
37                 end = mine;
38             else { answer = mine; break; }
39         }
40
41         cout << V(answer) << endl;
42     }
43
44     return 0;
45 }

```

## uva/12192.cpp

```

1  //12192
2  //Grapevine
3  //Misc;Binary Search
4  #include <iostream>
5  #include <cstring>
6  #include <vector>
7  #include <algorithm>
8  using namespace std;
9
10 int T[1001][501];
11 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-- > 0) {

```

```

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
2 //Smallest Regular Polygon
3 //Math;Geometry
4 #include <iostream>
5 #include <cmath>
6 #include <iomanip>
7 #define PI 3.141592653589793238462
8 using namespace std;
9
10 double cot(double angle) {
11     return cos(angle)/sin(angle);
12 }
13
14 int main(){
15     int x1, y1, x2, y2, n;
16     while(cin >> x1 >> y1 >> x2 >> y2 >> n, x1 | y1 | x2 | y2 | n) {
17         double d = sqrt(pow(x2-x1, 2.0)+pow(y2-y1, 2.0));

```

```

18         int k = n/2;
19         double s = sin(PI/n)/sin(PI*k/n)*d;
20         double A = 0.25*n*s*s*cot(PI/n);
21         setprecision(6);
22         cout << fixed << A << endl;
23     }
24 }
25 }

```

## uva/12361.cpp

```

1  //12361
2  //File Retrieval
3  //Misc;String Matching;Suffix Array;Longest Common Prefix
4  #include <iostream>
5  #include <iomanip>
6  #include <cstring>
7  #include <string>
8  #include <sstream>
9  #include <cmath>
10 #include <set>
11 #include <stack>
12 #define MAX 600200
13 #define ull unsigned long long
14 using namespace std;
15
16 struct Item {
17     ull v; int p;
18     Item(ull v, int p) : v(v), p(p) { }
19 };
20
21 int RA[MAX], tempRA[MAX];
22 int SA[MAX], tempSA[MAX];
23 int C[MAX];
24 int Phi[MAX], PLCP[MAX], LCP[MAX];
25 int IDX[MAX], SIZ[MAX];
26 set<ull> R;
27
28 void suffix_sort(int n, int k) {
29     memset(C, 0, sizeof C);
30
31     for (int i = 0; i < n; i++)
32         C[i + k < n ? RA[i + k] : 0]++;
33
34     int sum = 0;
35     for (int i = 0; i < max(256, n); i++) {
36         int t = C[i];
37         C[i] = sum;
38         sum += t;
39     }
40
41     for (int i = 0; i < n; i++)
42         tempSA[C[SA[i] + k < n ? RA[SA[i] + k] : 0]++] = SA[i];
43
44     memcpy(SA, tempSA, n*sizeof(int));
45 }
46
47 void suffix_array(string &s) {
48     int n = s.size();
49
50     for (int i = 0; i < n; i++)
51         RA[i] = s[i] - 1;
52
53     for (int i = 0; i < n; i++)
54         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())
137                     ST.top().v |= item.v;
138
139                 lastv = item.v;
140             }
141             if (LCP[i]) {
142                 if (ST.empty() || ST.top().p < LCP[i]) {
143                     ST.push(Item(1ull << IDX[SA[i]] | 1ull << IDX[SA[i-1]] | lastv, LCP[i]));
144                 } else if (ST.top().p == LCP[i]) {
145                     ST.top().v |= 1ull << IDX[SA[i]];
146                 }
147             }
148         }
149
150         while(!ST.empty()) {
151             Item item = ST.top(); ST.pop();
152
153             R.insert(item.v);
154             if (!ST.empty())
155                 ST.top().v |= item.v;
156         }

```

```

157     }
158
159
160     cout << R.size() << endl;
161 }
162 }

```

## uva/12363.cpp

```

1  //12363
2  //Hedge Mazes
3  //Graphs;Finding Bridges
4  #include <iostream>
5  #include <cstring>
6  #include <string>
7  #include <sstream>
8  #include <vector>
9  #include <algorithm>
10 #define MAX 10001
11 using namespace std;
12
13 int V[MAX], L[MAX], P[MAX], n, gpe;
14 vector<int> G[MAX];
15
16 inline int findset(int v) {
17     if (P[v] != -1 && P[v] != v)
18         return P[v] = findset(P[v]);
19     return v;
20 }
21
22 inline int unionset(int x, int y) {
23     int a = findset(x), b = findset(y);
24     if (a < b) swap(a, b);
25     P[b] = a;
26 }
27
28 void dfs(int u, int v) {
29     V[v] = L[v] = ++gpe;
30
31     for(int i = 0; i < G[v].size(); i++) {
32         int w = G[v][i];
33         if(!V[w]){
34             dfs(v, w);
35             L[v] = min(L[v], L[w]);
36
37             if (L[w] > V[v])
38                 unionset(v, w);
39         } else if(w != u) {
40             L[v] = min(L[v], V[w]);
41         }
42     }
43 }
44
45 int main() {
46     int m, q;
47     while(cin >> n >> m >> q, n|m|q) {
48         memset(G, 0, sizeof(vector<int>)*(n+1));
49         memset(V, 0, sizeof(int)*(n+1));
50         memset(L, 0, sizeof(int)*(n+1));
51         memset(P, -1, sizeof(int)*(n+1));
52         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
59     ull q = a/b;
60     ull x, y;
61     ull g = euclid(b, a-q*b, x, y);
62     return rx=y, ry=x-q*y, g;
63 }
64
65 ull invert(ull a, ull p) {
66     ull inverse, temp;
67     euclid(a, p, inverse, temp);
68     return inverse;
69 }
70
71 Fenwick T;
72
73 int main() {
74     int B, P, L, N;
75     while(cin >> B >> P >> L >> N, B|P|L|N) {
76         T.clear(L);
77         for(int i=0; i<N; i++) {
78             char cmd; ull a, b;
79             cin >> cmd >> a >> b;
80             if (cmd == 'E') {
81                 T.update(a, b*pow(B, L-a, P), P);
```

```

82         } else {
83             ull raw = T.rsq(a, b, P);
84             ull base = pow(B, L-b, P);
85
86             cout << ((raw*invert(base, P))%P+P)%P << endl;
87         }
88     }
89     cout << "-" << endl;
90 }
91
92 return 0;
93 }

```

## uva/12482.cpp

```

1  //12482
2  //Short Story Competition
3  //Misc;Ad hoc
4  #include <iostream>
5  #include <cmath>
6  #include <string>
7  using namespace std;
8
9  int main() {
10     int n, li, c;
11     while(cin >> n >> li >> c) {
12         string s;
13         int pag = 1;
14         int car = 0;
15         for(int i=0; i<n; i++) {
16             if (car > 0) car++;
17
18             cin >> s;
19             if (car + s.size() > c) {
20                 pag++;
21                 car = s.size();
22             } else {
23                 car += s.size();
24             }
25             //cout << s << " " << pag << " " << car << endl;
26         }
27         cout << ceil(pag/(double)li) << endl;
28     }
29 }
30
31 }
32 }

```

## uva/12483.cpp

```

1  //12483
2  //Toboggan of Marbles
3  //Math;Geometry;Point to Line
4  #include <iostream>
5  #include <cmath>
6  #include <string>
7  #include <cstring>
8  #include <iomanip>
9  #define ull unsigned long long;
10 using namespace std;
11
12 struct Point {
13     double x, y;
14
15     Point() {}
16     Point(double x, double y) : x(x), y(y) {}
17
18     double dist(Point A) {
19         return sqrt(pow(A.x-x,2)+pow(A.y-y,2));
20     }
21
22     double toLine(Point A, Point B) {
23         double scale = ((x - A.x) * (B.x - A.x) + (y-A.y)*(B.y-A.y)) /
24             ((B.x - A.x) * (B.x - A.x) + (B.y-A.y)*(B.y-A.y));
25
26         return dist(Point(A.x + scale*(B.x-A.x), A.y + scale * (B.y-A.y)));
27     }
28
29     double toSegment(Point A, Point B) {
30         if ((x - A.x) * (B.x - A.x) + (y-A.y)*(B.y-A.y) <= 1e-6)
31             return dist(A);

```

```

32 |
33 |         if ((x - B.x) * (A.x - B.x) + (y-B.y)*(A.y-B.y) <= 1e-6)
34 |             return dist(B);
35 |
36 |         return toLine(A, B);
37 |     }
38 | };
39 |
40 | int main() {
41 |     int n, L, H;
42 |     while(cin >> n >> L >> H) {
43 |         Point pa, pb;
44 |         int ya, yb;
45 |
46 |         double minn = 100000000.0;
47 |         for(int i=0; i<n; i++) {
48 |             cin >> ya >> pa.x >> pa.y;
49 |
50 |             int lado = i&1?L:0;
51 |             int outroLado = i&1?0:L;
52 |
53 |             // cout << " " << pa.x << " " << pa.y << "^" << outroLado;
54 |             minn = min(minn, pa.toSegment(Point(outroLado, 0), Point(outroLado, H)));
55 |
56 |             //cout << " " << pa.toSegment(Point(outroLado, 0), Point(outroLado, H));
57 |             if (i>0) {
58 |                 minn = min(minn, pb.toSegment(Point(lado, ya), pa));
59 |                 // cout << " " << pb.toSegment(Point(lado, ya), pa);
60 |             }
61 |             //cout << endl;
62 |
63 |
64 |             yb = ya;
65 |             pb = pa;
66 |         }
67 |         cout << fixed << setprecision(2) << minn << endl;
68 |     }
69 | }
70 |
71 | }

```

## uva/12484.cpp

```

1 | //12484
2 | //Cards
3 | //Dynamic Programming;Minimax
4 | #include <iostream>
5 | #include <cstring>
6 | #include <algorithm>
7 | #define ull long long
8 | using namespace std;
9 |
10 | ull T[10006], Q[10006], M[10006];
11 |
12 |
13 | int main() {
14 |     int n;
15 |     while(cin >> n) {
16 |         for(int i=1; i<=n; i++) {
17 |             cin >> M[i];
18 |             M[i]+=M[i-1];
19 |         }
20 |
21 |         memset(T, 0, sizeof(ull)*n);
22 |
23 |         for(int i=1; i<=n; i++) {
24 |             for(int j=0; j<=n-i; j++)
25 |                 Q[j] = M[j+i]-M[j] - min(T[j], T[j+1]);
26 |
27 |             swap(T, Q);
28 |         }
29 |
30 |         cout << T[0] << endl;
31 |     }
32 | }
33 | }

```

## uva/12485.cpp

```

1 | //12485
2 | //Perfect Choir

```

```

3 //Misc;Ad hoc
4 #include <iostream>
5 #include <cmath>
6 #include <cstring>
7 using namespace std;
8
9 int T[10005];
10
11 int main() {
12     int n;
13     while(cin >> n) {
14         memset(T, 0, sizeof(T));
15         int total = 0;
16         for(int i=0; i<n; i++) {
17             int a; cin >> a;
18             total += a;
19             T[i] = a;
20         }
21
22         if (total % n != 0) {
23             cout << -1 << endl;
24             continue;
25         }
26
27         int media = total / n;
28         int maior = 0;
29         for(int i=0; i<n; i++) {
30             if (T[i] > media)
31                 maior += T[i] - media;
32         }
33
34         cout << maior +1 << endl;
35     }
36 }

```

## uva/12486.cpp

```

1 //12486
2 //Space Elevator
3 //Misc;Binary Search
4 #include <iostream>
5 #include <cmath>
6 #include <string>
7 #include <cstring>
8 #include <iomanip>
9 #include <vector>
10 #include <algorithm>
11 #include <cstdio>
12 #define ull unsigned long long int
13 using namespace std;
14
15 ull T[20][10];
16
17 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
2  //Midnight Cowboy
3  //Graphs;Markov Chain
4  #include <iostream>
5  #include <cmath>
6  #include <string>
7  #include <cstring>
8  #include <iomanip>
9  #define ull unsigned long long;
10 using namespace std;
11
12 int G[101][101], S[101];
13 double M[101], Q[101];
14
15 int main() {
16     int n, a, b, c;
17     while(cin >> n >> a >> b >> c) {
18         memset(M, 0, sizeof(M));
19         memset(S, 0, sizeof(S));
20         memset(G, 0, sizeof(G));
21
22         for(int i=0; i<n-1; i++) {
23             int a, b; cin >> a >> b;
24             G[a][S[a]++] = b;
25             G[b][S[b]++] = a;
26         }
27
28         M[a] = 1.0;
29         for(int k=0; k<10000; k++) {
30             memset(Q, 0, sizeof(Q));
31             Q[b] = M[b];
32             Q[c] = M[c];

```

```

33         for(int i=1;i<=n;i++) {
34             //          cout << M[i] << " ";
35             if (i==b || i==c) continue;
36
37             for(int j=0;j<S[i];j++)
38                 Q[G[i][j]] += M[i] * 1.0/S[i];
39         }
40         //          cout << endl;
41
42         swap(Q, M);
43     }
44
45     cout << fixed << setprecision(6) << M[b] << endl;
46 }
47 }

```

## uva/12488.cpp

```

1 //12488
2 //Start Grid
3 //Misc;Ad hoc
4 #include <iostream>
5 #include <cstring>
6 #include <cmath>
7 using namespace std;
8
9 int L[30], C[30];
10
11 int abs(int n) {
12     if (n < 0) return -n;
13     return n;
14 }
15
16 int main() {
17     int n;
18     while(cin >> n) {
19         for(int i=0; i<n; i++)
20             cin >> L[i];
21
22         for(int i=0; i<n; i++)
23             cin >> C[i];
24
25         int total = 0;
26         for(int i=0; i<n; i++) {
27             int cara = C[i];
28             for(int j=n-1; j>i; j--) {
29                 if (L[j]==cara) {
30                     int t = L[j];
31                     L[j] = L[j-1];
32                     L[j-1] = t;
33                     total++;
34                 }
35             }
36         }
37
38         cout << total << endl;
39     }
40 }

```

## uva/12489.cpp

```

1 //12489
2 //Combating cancer
3 //Graphs;Tree Isomorphism
4 #include <iostream>
5 #include <cmath>
6 #include <string>
7 #include <cstring>
8 #include <iomanip>
9 #include <vector>
10 #include <algorithm>
11 #define MAX 10006
12 using namespace std;
13
14 vector<int> A[MAX], B[MAX];
15 vector<int> NA[MAX], NB[MAX];
16
17 bool comp(const vector<int>& a, const vector<int>& b) {
18     if (a.size() != b.size()) return a.size() < b.size();
19     for(int i=0;i<a.size(); i++) {
20         if (a[i] != b[i]) return a[i] < b[i];

```

```

21     }
22     return false;
23 }
24
25 bool eq(const vector<int>& a, const vector<int>& b) {
26     if (a.size() != b.size()) return false;
27     for(int i=0; i<a.size(); i++) {
28         if (a[i] != b[i]) return false;
29     }
30     return true;
31 }
32
33 int main() {
34     int n;
35     while(cin >> n) {
36         memset(A, 0, sizeof(A));
37         memset(B, 0, sizeof(B));
38         memset(NA, 0, sizeof(NA));
39         memset(NB, 0, sizeof(NB));
40         for(int i=0; i<n-1; i++) {
41             int a, b; cin >> a >> b;
42             A[a].push_back(b);
43             A[b].push_back(a);
44         }
45         for(int i=0; i<n-1; i++) {
46             int a, b; cin >> a >> b;
47             B[a].push_back(b);
48             B[b].push_back(a);
49         }
50
51
52         for(int i=1; i<=n; i++) {
53             for(int j=0; j<A[i].size(); j++)
54                 NA[i].push_back(A[A[i][j]].size());
55             sort(NA[i].begin(), NA[i].end());
56
57             for(int j=0; j<B[i].size(); j++)
58                 NB[i].push_back(B[B[i][j]].size());
59             sort(NB[i].begin(), NB[i].end());
60         }
61
62         sort(NA+1, NA+n+1, comp);
63         sort(NB+1, NB+n+1, comp);
64
65
66         bool equals = true;
67         //cout << NA[n].size() << " " << NA[n].size() << " " << n << endl;
68         for(int i=1; i<=n; i++) {
69             // cout << i << " => ";
70             // for(int j=0; j<NA[i].size(); j++) {
71                 // cout << NA[i][j] << " ";
72             // }
73             // cout << " | ";
74             // for(int j=0; j<NB[i].size(); j++) {
75                 // cout << NB[i][j] << " ";
76             // }
77             // cout << endl;
78
79             equals &= eq(NA[i], NB[i]);
80         }
81         cout << (equals ? "S" : "N") << endl;
82     }
83 }
84 }

```

## uva/12490.cpp

```

1 //12490
2 //Integral
3 //Misc;Ad hoc
4 #include <iostream>
5 #include <string>
6 #include <algorithm>
7 #include <cmath>
8 #define MAX 1000006
9 #define ull long long
10 using namespace std;
11
12 struct Value {
13     int x;
14     ull v;
15
16     inline bool operator <(const Value& a) const {

```

```

17         return this->x < a.x;
18     }
19 };
20
21 Value F[MAX];
22
23 int main() {
24     int n, s, y;
25     while(cin >> n >> s >> y) {
26         for(int i=0; i<s; i++) {
27             cin >> F[i].x >> F[i].v;
28         }
29
30         sort(F, F+s);
31
32         double minn = 0, maxx = 0;
33
34         for(int i=0; i<s-1; i++) {
35             Value a = F[i], b = F[i+1];
36
37             minn += min(a.v, b.v) + (b.x - a.x-1) * min(a.v, b.v) + abs(a.v - b.v)/2.0;
38             maxx += min(a.v, b.v) + (b.x - a.x-1) * max(a.v, b.v) + abs(a.v - b.v)/2.0;
39         }
40
41         if (y < minn || y > maxx || ceil(minn) != minn) {
42             cout << "N" << endl; continue;
43         }
44
45         cout << "S";
46
47         ull current = round(maxx);
48         for(int i=0; i<s-1; i++) {
49             Value a = F[i], b = F[i+1];
50
51             ull delta = (b.x - a.x-1) * (max(a.v, b.v) - min(a.v, b.v));
52
53             if (current == y) {
54                 for(int x = a.x+1; x<b.x; x++)
55                     cout << " " << max(a.v, b.v);
56             } else if (current - delta > y) {
57                 current -= delta;
58
59                 for(int x = a.x+1; x<b.x; x++)
60                     cout << " " << min(a.v, b.v);
61             } else if (a.v < b.v) {
62                 for(int x=a.x+1; x<b.x; x++) {
63                     ull value = max(a.v, b.v - (current - y));
64                     cout << " " << value;
65                     current -= b.v - value;
66                 }
67             } else {
68                 ull plus = (current - y) / (b.x - a.x - 1);
69                 ull rem = (current - y) % (b.x - a.x - 1);
70                 for(int x=a.x+1; x<b.x; x++) {
71                     ull value = a.v - plus - (b.x - x <= rem ? 1: 0);
72                     cout << " " << value;
73                     current -= a.v - value;
74                 }
75             }
76         }
77         cout << endl;
78     }
79 }
80 }

```

## uva/12491.cpp

```

1 //12491
2 //Words
3 //Misc;STL map
4 #include <iostream>
5 #include <string>
6 #include <map>
7 #include <set>
8 #define mit multimap<string, string>::iterator
9 #define mmit pair<multimap<string, string>::iterator, multimap<string, string>::iterator>
10 using namespace std;
11
12 multimap<string, string> X[2];
13 set<string> S[2], E[2];
14 char T[1000];
15
16 bool backtrack(int x, int k, int n) {

```

```

17     string suffix(T+k, n-k);
18     if (S[x].find(suffix) != S[x].end()) return false;
19     S[x].insert(suffix);
20
21     for(int s=1; s<=suffix.size(); s++) {
22         string word = suffix.substr(0, s);
23
24         for(int i=0; i<word.size(); i++)
25             T[k+i] = word[i];
26
27         if (E[x].find(word) != E[x].end()) {
28             if (k + word.size()==n) return true;
29             if (backtrack(x, k+word.size(), n)) return true;
30         }
31     }
32 }
33
34 mmit ret = X[x].equal_range(suffix);
35 for(mit it=ret.first;it != ret.second;it++) {
36     string word = it->second;
37     for(int i=0; i<word.size(); i++)
38         T[k+i] = word[i];
39
40     if (k + word.size() == n) return true;
41     if (backtrack(1-x, n, k+word.size())) return true;
42 }
43 return false;
44 }
45
46 int main() {
47     int a, b;
48     while(cin >> a >> b) {
49         E[0].clear(); E[1].clear();
50         X[0].clear(); X[1].clear();
51         S[0].clear(); S[1].clear();
52         for(int i=0; i<a; i++) {
53             string s; cin >> s;
54             for(int j=0; j<=s.size(); j++) {
55                 E[0].insert(s);
56                 X[0].insert(pair<string, string>(s.substr(0, j), s));
57             }
58         }
59         for(int i=0; i<b; i++) {
60             string s; cin >> s;
61             for(int j=0; j<=s.size(); j++) {
62                 E[1].insert(s);
63                 X[1].insert(pair<string, string>(s.substr(0, j), s));
64             }
65         }
66         cout << (backtrack(0,0,0) || backtrack(1,0,0) ? 'S' : 'N') << endl;
67     }
68 }
69 }

```

## uva/12492.cpp

```

1 //12492
2 //Rubik Cycle
3 //Misc;Ad hoc
4 #include <iostream>
5 #include <string>
6 #define MAX 200010
7 using namespace std;
8
9 int T[54];
10
11 void rotate(int a, int b, int c, int d, int e, int f, int g, int h) {
12     int x = T[h], y = T[g];
13     T[h] = T[f];
14     T[g] = T[e];
15
16     T[f] = T[d];
17     T[e] = T[c];
18
19     T[d] = T[b];
20     T[c] = T[a];
21
22     T[b] = x;
23     T[a] = y;
24 }
25
26 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) {
27     int x = T[j], y = T[k], z = T[l];

```

```

28
29     T[j] = T[g];
30     T[k] = T[h];
31     T[l] = T[i];
32
33     T[g] = T[d];
34     T[h] = T[e];
35     T[i] = T[f];
36
37     T[d] = T[a];
38     T[e] = T[b];
39     T[f] = T[c];
40
41     T[a] = x;
42     T[b] = y;
43     T[c] = z;
44 }
45
46 void F() {
47     rotate(0, 1, 2, 5, 8, 7, 6, 3);
48     adjust(33, 34, 35, 45, 48, 51, 11, 10, 9, 44, 41, 38);
49 }
50
51 void B() {
52     rotate(26, 25, 24, 21, 18, 19, 20, 23);
53     adjust(29, 28, 27, 36, 39, 42, 15, 16, 17, 53, 50, 47);
54 }
55
56 void L() {
57     rotate(36, 37, 38, 41, 44, 43, 42, 39);
58     adjust(0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33);
59 }
60
61 void R() {
62     rotate(45, 46, 47, 50, 53, 52, 51, 48);
63     adjust(8, 5, 2, 35, 32, 29, 26, 23, 20, 17, 14, 11);
64 }
65
66 void U() {
67     rotate(27, 28, 29, 32, 35, 34, 33, 30);
68     adjust(2, 1, 0, 38, 37, 36, 24, 25, 26, 47, 46, 45);
69 }
70
71 void D() {
72     rotate(9, 10, 11, 14, 17, 16, 15, 12);
73     adjust(6, 7, 8, 51, 52, 53, 20, 19, 18, 42, 43, 44);
74 }
75
76
77 bool ok() {
78     for(int i=0; i<54; i++) {
79         if (T[i] != i) return false;
80     }
81     return true;
82 }
83
84 int main() {
85     string s;
86     while(cin >> s) {
87         for(int i=0; i<54; i++)
88             T[i] = i;
89
90         int result = 0;
91         do {
92             for(int i=0; i<s.size(); i++) {
93                 switch(s[i]) {
94                     case 'F': F(); break;
95                     case 'B': B(); break;
96                     case 'R': R(); break;
97                     case 'L': L(); break;
98                     case 'U': U(); break;
99                     case 'D': D(); break;
100                    case 'f': F(); F(); F(); break;
101                    case 'b': B(); B(); B(); break;
102                    case 'r': R(); R(); R(); break;
103                    case 'l': L(); L(); L(); break;
104                    case 'u': U(); U(); U(); break;
105                    case 'd': D(); D(); D(); break;
106                }
107            }
108            result++;
109        } while (!ok());
110        cout << result << endl;
111    }

```

## uva/12493.cpp

```

1  //12493
2  //Stars
3  //Math;Prime Factorization;Euler's Totient
4  #include <iostream>
5  #include <cmath>
6  #define PP 100000
7  #define ull unsigned long long
8  using namespace std;
9
10 bool P[PP];
11
12 int main() {
13     for(long long i=2; i<PP; i++) {
14         if (P[i]) continue;
15         for(long long j=i*i; j<PP; j+=i) {
16             P[j] = true;
17         }
18     }
19
20     ull n;
21     while(cin >> n) {
22         ull tot = 1;
23         //cout << ">>" << n << endl;
24         for(ull i=2; i*i<=n && n>1; i++) {
25             if (P[i]) continue;
26             ull q=0;
27
28             while(n%i==0) {
29                 n/=i;
30                 q++;
31             }
32
33             //if (q>0)
34             //cout << i << " " << q << endl;
35             if (q>0)
36                 tot *= (i-1) * (ull)pow(i, q-1);
37         }
38         if (n>1)
39             tot *= n-1;
40
41         cout << tot/2 << endl;
42     }
43 }
44

```

## uva/12506.cpp

```

1  //12506
2  //Shortest Names
3  //Misc;String Matching;Trie
4  #include <iostream>
5  #include <cstring>
6  #define MAXS 1000010
7  using namespace std;
8
9  struct Trie {
10     int G[MAXS][26];
11     int S[MAXS];
12     int stateCount;
13
14     Trie() {
15         clear();
16     }
17
18     void clear() {
19         stateCount = 0;
20         clear(stateCount++);
21     }
22
23     int clear(int state) {
24         memset(G[state], -1, sizeof G[state]);
25         S[state] = 0;
26         return state;
27     }
28
29     void add(string &s) {
30         int state = 0;
31

```

```

31         for(int i=0; i<s.size(); i++) {
32             S[state]++;
33
34             int next = s[i] - 'a';
35
36             if (G[state][next] < 0)
37                 G[state][next] = clear(stateCount++);
38
39             state = G[state][next];
40         }
41     }
42 };
43
44 Trie T;
45
46 int dfs(int state) {
47     if (T.S[state] == 1) return 0;
48
49     int s = T.S[state];
50
51     for (int e = 0; e < 26; ++e) {
52         if (T.G[state][e] == -1) continue;
53
54         s += dfs(T.G[state][e]);
55     }
56
57     return s;
58 }
59
60
61 int main() {
62     int tt; cin >> tt;
63     while(tt--) {
64         T.clear();
65
66         int n; cin >> n;
67         for(int i=0; i<n; i++) {
68             string s; cin >> s;
69             T.add(s);
70         }
71
72         cout << dfs(0) << endl;
73     }
74 }

```

## timus/1017.cpp

```

1  //1017
2  //Staircases
3  //Dynamic Programming;Ad hoc
4  #include <iostream>
5  #define MAX 506
6  using namespace std;
7
8  long long T[MAX][MAX];
9
10 int main() {
11     for(int i=1; i<MAX; i++) {
12         for(int j=1; j<=i; j++) {
13             T[i][j] = 1;
14             for(int k=j; k<=i; k++)
15                 T[i][j] += T[i-k][k+1];
16         }
17     }
18
19     int n;
20     while(cin >> n)
21         cout << T[n][1]-1 << endl;
22 }

```

## timus/1018.cpp

```

1  //1018
2  //Binary Apple Tree
3  //Dynamic Programming;Ad hoc
4  #include <iostream>
5  #include <vector>
6  #include <cstring>
7  #define MAX 105
8  using namespace std;
9

```



```

10 struct Node {
11     int x, v;
12     Node(int x, int v) : x(x), v(v) {}
13 };
14 vector<Node> T[MAX];
15 int S[MAX], TT[MAX][MAX];
16 bool V[MAX][MAX];
17
18 void adjust(int root, int parent) {
19     for(int i=0; i<T[root].size(); i++) {
20         Node node = T[root][i];
21         if (node.x != parent) {
22             adjust(node.x, root);
23             S[node.x] = node.v;
24         } else {
25             T[root].erase(T[root].begin()+i);
26             i--;
27         }
28     }
29 }
30
31 int answer(int root, int branches) {
32     if (V[root][branches])
33         return TT[root][branches];
34
35     if (branches == 0) return 0;
36     if (branches == 1) return S[root];
37     if (T[root].size() == 0) return S[root];
38     if (T[root].size() == 1) return S[root] + answer(T[root][0].x, branches-1);
39
40     Node left = T[root][0];
41     Node right = T[root][1];
42
43     int maxx = 0;
44     for(int i=0; i<=branches-1; i++)
45         maxx = max(maxx, S[root] + answer(left.x, i) + answer(right.x, branches-1-i));
46
47     V[root][branches] = true;
48     return TT[root][branches] = maxx;
49 }
50
51 int main() {
52     int n, q;
53     while(cin >> n >> q) {
54         memset(T, 0, sizeof(vector<int>)*n);
55         memset(V, 0, sizeof(vector<int>)*n);
56         memset(S, 0, sizeof(S));
57
58         for(int i=0; i<n-1; i++) {
59             int a, b, c; cin >> a >> b >> c;
60             T[b].push_back(Node(a,c));
61             T[a].push_back(Node(b,c));
62         }
63         adjust(1, -1);
64         cout << answer(1, q+1) << endl;
65     }
66 }

```

## timus/1020.cpp

```

1 //1020
2 //Rope
3 //Math;Geometry
4 #include <iostream>
5 #include <cmath>
6 #include <iomanip>
7 #define PI 3.14159265
8 using namespace std;
9
10 struct Point {
11     double x, y;
12
13     Point() {}
14
15     double dist(Point A) {
16         return sqrt(pow(A.x-x,2)+pow(A.y-y,2));
17     }
18 };
19
20 int main() {
21     int n;
22     double r;
23     Point a, b, c;

```

```

24     while(cin >> n >> r) {
25         double total = 0;
26
27         cin >> a.x >> a.y;
28         b = a;
29         for(int i=1; i<n; i++) {
30             cin >> c.x >> c.y;
31             total += b.dist(c);
32             swap(b, c);
33         }
34         total += a.dist(b);
35
36         cout << fixed << setprecision(2) << total + 2*PI*r << endl;
37     }
38 }

```

## timus/1030.cpp

```

1  //1030
2  //Titanic
3  //Math;Geometry;Great-Circle Distance
4  #include <iostream>
5  #include <cmath>
6  #include <string>
7  #include <iomanip>
8  #define PI 3.14159265
9  using namespace std;
10
11 double distance(double r, double x1, double y1, double x2, double y2) {
12     return r*acos(sin(x1)*sin(x2) + cos(x1)*cos(x2)*cos(fabs(y1-y2)));
13 }
14
15 double readCoord() {
16     int a1, a2, a3; char c; string s;
17     cin >> a1 >> c >> a2 >> c >> a3 >> c >> s;
18     double ret = (a1 + a2/60.0 + a3/3600.0) / 180.0 * PI;
19     if (s=="WL." || s=="SL")
20         ret = -ret;
21     return ret;
22 }
23
24 double round(double d)
25 {
26     return floor(d + 0.5);
27 }
28
29 int main() {
30     string s;
31
32     getline(cin, s);
33     getline(cin, s);
34     getline(cin, s);
35
36     double X1 = readCoord();
37     cin >> s;
38     double Y1 = readCoord();
39     getline(cin, s);
40     getline(cin, s);
41     double X2 = readCoord();
42     cin >> s;
43     double Y2 = readCoord();
44
45     double d = distance(6875.0/2, X1, Y1, X2, Y2);
46     cout << "The distance to the iceberg: " << fixed << setprecision(2) << round(d*100)/100.0 << " miles." << endl;
47     if (d < 99.995)
48         cout << "DANGER!" << endl;
49
50 }

```

## timus/1111.cpp

```

1  //1111
2  //Squares
3  //Math;Geometry;Square Distance
4  #include <iostream>
5  #include <cmath>
6  #include <vector>
7  #include <algorithm>
8  using namespace std;
9

```

```

10 struct Point {
11     double x, y;
12
13     Point() {}
14     Point(double x, double y) : x(x), y(y) {}
15
16     double dist(const Point A) const {
17         return sqrt(pow(A.x-x,2)+pow(A.y-y,2));
18     }
19
20     double toLine(const Point A, const Point B) const {
21         double scale = ((x - A.x) * (B.x - A.x) + (y-A.y)*(B.y-A.y)) /
22             ((B.x - A.x) * (B.x - A.x) + (B.y-A.y)*(B.y-A.y));
23
24         return dist(Point(A.x + scale*(B.x-A.x), A.y + scale * (B.y-A.y)));
25     }
26
27     double toSegment(const Point A, const Point B) const {
28         if ((x - A.x) * (B.x - A.x) + (y-A.y)*(B.y-A.y) <= 1e-6)
29             return dist(A);
30
31         if ((x - B.x) * (A.x - B.x) + (y-B.y)*(A.y-B.y) <= 1e-6)
32             return dist(B);
33
34         return toLine(A, B);
35     }
36
37     int signal(const Point& a, const Point& b) const {
38         double sig = (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x);
39         if (abs(sig) < 1e-6) return 0;
40         if (sig < 0) return -1;
41         return 1;
42     }
43
44     Point rotateWith(const Point origin, double si, double co, double scale) const {
45         double tx = this->x - origin.x;
46         double ty = this->y - origin.y;
47         double x = (tx * co + ty * si)/scale;
48         double y = (tx * -si + ty * co)/scale;
49         return Point(origin.x + x, origin.y + y);
50     }
51 }
52 };
53
54 struct Square {
55     int id;
56     Point a, b, c, d;
57     Square(int id, Point x, Point y) : id(id),
58         a(x.x, x.y), b(x.rotateWith(y, 0.707106781, 0.707106781, 1.41421356)),
59         c(y.x, y.y), d(x.rotateWith(y, -0.707106781, 0.707106781, 1.41421356)) {}
60
61     bool inside(const Point p) const {
62         int sig = a.signal(p, b);
63         if (sig == 0) return false;
64         if (sig != b.signal(p, c)) return false;
65         if (sig != c.signal(p, d)) return false;
66         if (sig != d.signal(p, a)) return false;
67         return true;
68     }
69
70     double dist(const Point p) const {
71         if (inside(p)) return 0.0;
72         return min(min(p.toSegment(a,b), p.toSegment(b, c)), min(p.toSegment(c,d), p.toSegment(d,a)));
73     }
74 };
75
76 struct DistToP {
77     Point p;
78     DistToP(Point p) : p(p) {}
79
80     inline bool operator() (const Square &a, const Square &b) {
81         double da = a.dist(this->p), db=b.dist(this->p);
82         if (abs(da-db) > 1e-6) return da<db;
83         return a.id < b.id;
84     }
85 };
86
87 vector<Square> V;
88
89 int main() {
90     int n;
91     cin >> n;
92     for(int i=1; i<=n; i++) {
93         Point x,y; cin >> x.x >> x.y >> y.x >> y.y;

```

```

94         V.push_back(Square(i, x, y));
95     }
96
97     Point p;
98     cin >> p.x >> p.y;
99
100    sort(V.begin(), V.end(), DistToP(p));
101    for(int i=0; i<n; i++) {
102        if (i) cout << " ";
103        cout << V[i].id;
104    }
105    cout << endl;
106 }

```

## timus/1159.cpp

```

1  //1185
2  //Wall
3  //Math;Geometry;Enclosing Circle
4  #include <iostream>
5  #include <cmath>
6  #include <iomanip>
7  #include <algorithm>
8  #define PI 3.14159265
9  #define EP 1e-10
10 using namespace std;
11
12 struct Point {
13     long double x, y;
14
15     Point() {}
16     Point(long double x, long double y) : x(x), y(y) {}
17 };
18
19 int P[107];
20
21 long double simpleCase(long double maxx, int n) {
22     long double begin=maxx/2.0, end=50000001.0;
23     while(abs(begin-end) > EP) {
24         long double r = (begin+end)/2;
25         long double angle = 0;
26         long double sum = 0;
27         Point A(r,0);
28         for(int i=0; i<n; i++) {
29             angle += 2*asin(P[i]/(2.0*r));
30             Point B(r*cos(angle), r*sin(angle));
31             sum += (A.x + B.x) * (B.y - A.y);
32             A = B;
33         }
34         sum /= 2;
35
36         if (abs(angle-2*PI) < 1e-4)
37             return sum;
38
39         if (angle < 2*PI)
40             end = r;
41         else
42             begin = r;
43     }
44     return 0.0;
45 }
46 long double complexCase(long double maxx, int n) {
47     long double begin=maxx/2.0, end=50000001.0;
48     while(abs(begin-end) > EP) {
49         long double r = (begin+end)/2;
50         long double angle = 2*asin(P[0]/(2.0*r));
51         Point A(r*cos(angle), r*sin(angle));
52         long double sum = (r + A.x) * (0 - A.y);
53         for(int i=1; i<n; i++) {
54             angle -= 2*asin(P[i]/(2.0*r));
55             Point B(r*cos(angle), r*sin(angle));
56             sum += (A.x + B.x) * (A.y - B.y);
57             A = B;
58         }
59         sum /= 2;
60
61         if (abs(angle) < EP)
62             return sum;
63
64         if (angle < 0)
65             end = r;
66         else
67             begin = r;

```

```

68     }
69     return 0.0;
70 }
71
72 bool comp(int a, int b) {
73     return a>b;
74 }
75
76 int main() {
77     int n;
78     while(cin >> n) {
79         int maxx = 0, summ = 0;
80         for(int i=0; i<n; i++) {
81             cin >> P[i];
82             maxx = max(maxx, P[i]);
83             summ += P[i];
84         }
85
86         sort(P, P+n, comp);
87         cout << fixed << setprecision(2) << max(simpleCase(maxx, n), complexCase(maxx, n)) << endl;
88
89     }
90 }

```

## timus/1185.cpp

```

1  //1185
2  //Wall
3  //Math;Geometry;Convex Hull;Monotone Chain
4  #include <iostream>
5  #include <cmath>
6  #include <iomanip>
7  #include <algorithm>
8  #define PI 3.14159265
9  using namespace std;
10
11 struct Point {
12     int x, y;
13
14     Point() {}
15     Point(int x, int y) : x(x), y(y) {}
16
17     bool left(Point& a, Point& b) {
18         return (this->x - a.x)*(b.y - a.y) - (this->y - a.y)*(b.x - a.x) < 0;
19     }
20
21     bool operator <(const Point& p) const {
22         if (this->x != p.x) return this->x < p.x;
23         return this->y < p.y;
24     }
25
26     bool operator ==(const Point& p) const {
27         return this->x == p.x and this->y == p.y;
28     }
29
30     double dist(Point A) {
31         return sqrt(pow(A.x-x,2.0)+pow(A.y-y,2.0));
32     }
33 };
34
35 int convexHull(Point* P, int n, Point* S) {
36     sort(P, P+n);
37
38     int m=0;
39     for(int i=0; i<n; i++) {
40         while(m >= 2 && S[m-1].left(S[m-2], P[i])) m--;
41         S[m++] = P[i];
42     }
43     m--;
44
45     for(int i=n-1, k=m; i >= 0; i--) {
46         while(m >= k+2 && S[m-1].left(S[m-2], P[i])) m--;
47         S[m++] = P[i];
48     }
49     m--;
50
51     return m;
52 }
53
54 Point P[1010], S[1010];
55
56 int main() {
57     int n, r;

```

```

58     while(cin >> n >> r) {
59         for(int i=0; i<n; i++)
60             cin >> P[i].x >> P[i].y;
61
62         int s = convexHull(P, n, S);
63
64         double total = 0;
65         for(int i=0; i<s; i++)
66             total += S[i].dist(S[(i+1)%n]);
67
68         cout << floor(total + 2*PI*r + 0.5) << endl;
69     }
70 }

```

## timus/1258.cpp

```

1  //1258
2  //Pool
3  //Math;Geometry;Mirror
4  #include <iostream>
5  #include <cmath>
6  #include <string>
7  #include <iomanip>
8  #define ull long long
9
10 using namespace std;
11
12 int main() {
13     ull W, D, a, b, c, d;
14     string s;
15     while(cin >> W >> D >> a >> b >> c >> d) {
16         cin >> s;
17         ull x=0, y=0;
18         ull sr=1<<30, sl=1<<30, sf=1<<30, sb=1<<30;
19
20         for(ull i=0; i<s.size(); i++) {
21             switch(s[i]) {
22                 case 'R': x+=2*(W-c); sr = min(sr, i); break;
23                 case 'L': x+=2*c; sl = min(sl, i); break;
24                 case 'F': y+=2*d; sf = min(sf, i); break;
25                 case 'B': y+=2*(D-d); sb = min(sb, i); break;
26             }
27         }
28
29         c += sr<sl?x:-x;
30         d += sb<sf?y:-y;
31
32         cout << fixed << setprecision(4) << sqrt((a-c)*(a-c)+(b-d)*(b-d)+0.0) << endl;
33
34     }
35 }

```

## timus/1332.cpp

```

1  //1332
2  //Genie Bomber
3  //Math;Geometry;Enclosing Circle
4  #include <iostream>
5  #include <cmath>
6  #define EP 1e-6
7  using namespace std;
8
9  struct Point {
10     double x, y;
11
12     Point() {}
13     Point(double x, double y) : x(x), y(y) {}
14
15     double dist(Point A) {
16         return sqrt(pow(A.x-x,2)+pow(A.y-y,2));
17     }
18
19     Point middle(Point B) {
20         return Point((x+B.x)/2, (y+B.y)/2);
21     }
22 };
23
24 struct Circle {
25     Point c; double r;
26     Circle(Point c, double r) : c(c), r(r) {}

```

```

28
29     Circle(Point p1, Point p2, double r) {
30         double d2 = (p1.x - p2.x) * (p1.x - p2.x) +
31                     (p1.y - p2.y) * (p1.y - p2.y);
32         double det = r*r / d2 - 0.25;
33         double h = sqrt(det);
34
35         this->c = Point((p1.x + p2.x) * 0.5 + (p1.y - p2.y) * h,
36                       (p1.y + p2.y) * 0.5 + (p2.x - p1.x) * h);
37         this->r = r;
38     }
39
40
41     static bool invalid(Point p1, Point p2, double r) {
42         double d2 = (p1.x - p2.x) * (p1.x - p2.x) +
43                     (p1.y - p2.y) * (p1.y - p2.y);
44         double det = r*r / d2 - 0.25;
45         return det < 0.0;
46     }
47
48     bool within(Point p) {
49         return c.dist(p)-r < EP;
50     }
51 };
52
53 Point P[106];
54
55 int best(Circle c1, int n) {
56     int sum1 = 0;
57     for(int k=0; k<n; k++) {
58         if (c1.within(P[k]))
59             sum1++;
60     }
61     return sum1;
62 }
63
64
65 int main() {
66     int n, r, R;
67     while(cin >> n) {
68         for(int i=0; i<n; i++)
69             cin >> P[i].x >> P[i].y;
70         cin >> R >> r;
71         R-=r;
72
73         int maxx = 0;
74         for(int i=0; i<n; i++) {
75             for(int j=0; j<n; j++) {
76                 if (i==j) {
77                     maxx = max(maxx, best(Circle(P[i], R), n));
78                 } else if (!Circle::invalid(P[i], P[j], R)) {
79                     maxx = max(maxx, best(Circle(P[i], P[j], R), n));
80                     maxx = max(maxx, best(Circle(P[i], P[j], R), n));
81                 }
82             }
83         }
84
85         cout << maxx << endl;
86     }
87 }
88

```

## timus/1373.cpp

```

1  //1373
2  //Pictura ex Machina
3  //Math;Geometry;Segment Rotation
4  #include <iostream>
5  #include <cmath>
6  #include <iomanip>
7  using namespace std;
8
9  double PI = 2*acos(0.0);
10
11 struct Point {
12     double x, y;
13
14     Point() {}
15     Point(double x, double y) : x(x), y(y) {}
16
17     Point rotateWith(const Point origin, double si, double co, double scale) const {
18         double tx = this->x - origin.x;
19         double ty = this->y - origin.y;
20

```

```

20     double x = (tx * co + ty * si)/scale;
21     double y = (tx * -si + ty * co)/scale;
22     return Point(origin.x + x, origin.y + y);
23
24 }
25 };
26
27 double round4(double a) {
28     if (a<0) return 0.0;
29     return floor(a*10000+0.5)/10000.0;
30 }
31
32 int main() {
33     Point a, b;
34     double minx=1<<30, maxx=0, miny=1<<30, maxy=0;
35     while(cin >> a.x >> a.y >> b.x >> b.y) {
36         Point c = b.rotateWith(a, sin(-PI/4), cos(-PI/4), sqrt(2.0));
37
38         minx = min(minx, min(min(a.x, b.x), c.x));
39         miny = min(miny, min(min(a.y, b.y), c.y));
40         maxx = max(maxx, max(max(a.x, b.x), c.x));
41         maxy = max(maxy, max(max(a.y, b.y), c.y));
42     }
43
44     cout << fixed << setprecision(4) << round4(maxx - minx) << " " << round4(maxy - miny) << endl;
45 }

```

## timus/1422.cpp

```

1  //1422
2  //Fireflies
3  //Math;Geometry;3D Line Detection
4  #include <iostream>
5  #include <cmath>
6  #include <map>
7  using namespace std;
8
9  int gcd(int a, int b) {
10     while(b)
11         swap(a=a%b,b);
12     return a;
13 }
14
15 struct Vector {
16     int x, y, z;
17
18     Vector() : x(0), y(0), z(0) {}
19     Vector(int x, int y, int z) : x(x), y(y), z(z) {}
20
21     Vector normalize() const {
22         int d = gcd(x, gcd(y, z));
23         return Vector(x/d, y/d, z/d);
24     }
25
26     Vector operator -(const Vector& that) const {
27         return Vector(x-that.x, y-that.y, z-that.z);
28     }
29
30     bool operator <(const Vector& that) const {
31         if (x!=that.x) return x<that.x;
32         if (y!=that.y) return y<that.y;
33         return z<that.z;
34     }
35 }
36
37 };
38
39 Vector P[2007];
40 map<Vector, int> M;
41
42 int main() {
43     int n;
44     while(cin >> n) {
45         for(int i=0; i<n; i++)
46             cin >> P[i].x >> P[i].y >> P[i].z;
47
48         int maxx = 0;
49         for(int i=0; i<n; i++) {
50             M.clear();
51             for(int j=i+1; j<n; j++)
52                 maxx = max(maxx, ++M[(P[i]-P[j]).normalize()]);
53         }
54     }

```



```

55 |
56 |         cout << maxx+1 << endl;
57 |     }
58 | }
59 |

```

## timus/1578.cpp

```

1  //1578
2  //Mammoth Hunt
3  //Math;Geometry;Segments Angle
4  #include <iostream>
5  #include <cmath>
6  #define EP 1e-6
7  #define PI 3.14159265
8  using namespace std;
9
10 struct Point {
11     int x, y;
12
13     Point() {}
14     Point(int x, int y) : x(x), y(y) {}
15
16     double dist(Point A) {
17         return sqrt(pow(A.x-x,2.0)+pow(A.y-y,2.0));
18     }
19
20     double angle(Point B, Point C) {
21         double a = dist(B), b = B.dist(C), c=dist(C);
22         double ret = acos((a*a+b*b-c*c) / (2*a*b));
23         if (ret < 0) ret += 2*PI;
24         //cout << " " << a << " " << b << " " << c << " " << ret << endl;
25         return ret;
26     }
27
28     bool accute(Point B, Point C) {
29         return angle(B, C) < PI/2.0;
30     }
31 };
32
33 Point P[2010];
34 int O[2010], V[2010];
35 int n;
36
37 bool dfs(int v, int i) {
38     O[i] = v; V[v] = i;
39     if (i==n) return true;
40
41     for(int j=1; j<=n; j++) {
42         if (V[j]) continue;
43         Point a = P[O[i-1]], b = P[O[i]], c = P[j];
44         if (a.accute(b,c))
45             if (dfs(j, i+1))
46                 return true;
47     }
48     V[v] = 0;
49
50     return false;
51 }
52
53 int main() {
54     int k;
55     while(cin >> k) {
56         memset(O, 0, sizeof(O));
57         memset(V, 0, sizeof(V));
58
59         n = k + 2;
60         for(int i=1; i<=n; i++)
61             cin >> P[i].x >> P[i].y;
62
63         bool ok = false;
64         for(int i=1; i<=n && !ok; i++) {
65             O[1] = i; V[i] = 1;
66             for(int j=1; j<=n && !ok; j++) {
67                 if (i==j) continue;
68                 ok |= dfs(j, 2);
69             }
70         }
71
72         if (ok) {
73             cout << "YES" << endl;
74         }
75     }

```

```

76         for(int i=1; i<=n; i++) {
77             if (i>1) cout << " ";
78             cout << 0[i];
79         }
80         cout << endl;
81     } else {
82         cout << "NO" << endl;
83     }
84
85 }
86
87 }

```

## timus/1658.cpp

```

1  //1658
2  //Sum of Digits
3  //Dynamic Programming;Ad hoc
4  #include <iostream>
5  #include <vector>
6  #include <cstring>
7  using namespace std;
8
9  short S[1010][9000];
10 short T[1010][9000];
11
12 int main() {
13     T[0][0] = 1;
14     for(int i=1; i<=1000; i++) {
15         for(int j=1; j<=8100; j++) {
16             S[i][j] = 102;
17             for(int k=1; k<=9; k++) {
18                 int a = i-k;
19                 int b = j-k*k;
20                 if (a>=0 && b>=0 && T[a][b] && S[a][b]+1<S[i][j]) {
21                     T[i][j] = k;
22                     S[i][j] = S[a][b]+1;
23                 }
24             }
25         }
26     }
27
28     int t; cin >> t;
29     while(t-->0) {
30         int s1, s2;
31         cin >> s1 >> s2;
32
33         if (s1 > 1000 || s2 > 8100 || S[s1][s2] > 100) {
34             cout << "No solution" << endl;
35             continue;
36         }
37
38         int n = S[s1][s2];
39         for(int i=0; i<n; i++) {
40             int d = T[s1][s2];
41             cout << d;
42             s1 -= d;
43             s2 -= d*d;
44         }
45         cout << endl;
46     }
47 }
48

```

## spoj/nkmobile.cpp

```

1  //NKMOBILE
2  //IOI01 Mobiles
3  //Misc;Fenwick Tree;2D
4  #include <iostream>
5  #include <cstring>
6  #define MAX 1030
7  using namespace std;
8
9  struct Fenwick2D {
10     int T[MAX][MAX];
11     int n, m;
12
13     void clear(int n, int m) {
14         for(int i=1; i<=n; i++)
15             for(int j=1; j<=m; j++)

```

```

16         T[i][j] = 0;
17
18         this->n = n;
19         this->m = m;
20     }
21
22     void adjust(int x, int y, int v) {
23         for (int i=x; i <= n; i += (i&-i))
24             for(int j=y; j <= m; j += (j&-j))
25                 T[i][j] += v;
26     }
27
28     int rsq(int x, int y) {
29         int sum = 0;
30         for(int i=x; i; i -= (i&-i))
31             for(int j=y; j; j -= (j&-j))
32                 sum += T[i][j];
33         return sum;
34     }
35
36     int rsq(int x1, int y1, int x2, int y2) {
37         return rsq(x2, y2) - rsq(x2, y1-1) - rsq(x1-1, y2) + rsq(x1-1, y1-1);
38     }
39 };
40
41 Fenwick2D T;
42
43 int main() {
44     int cmd;
45     while(cin >> cmd, cmd != 3) {
46         if (cmd == 0) {
47             int s;
48             cin >> s;
49             T.clear(s, s);
50         } else if (cmd == 1) {
51             int x, y, a;
52             cin >> x >> y >> a;
53             x++; y++;
54             T.adjust(x, y, a);
55         } else if (cmd == 2) {
56             int x1, y1, x2, y2;
57             cin >> x1 >> y1 >> x2 >> y2;
58             x1++; y1++; x2++; y2++;
59             cout << T.rsq(x1, y1, x2, y2) << endl;
60         }
61     }
62 }

```

## spojbr/homem.cpp

```

1  //HOMEM
2  //Homem, Elefante e Rato
3  //Misc;Segment Tree;Lazy Propagation
4  #include <iostream>
5  #include <cstring>
6  #include <cstdio>
7  #define MAX 600100
8  #define ull long long
9  using namespace std;
10
11 struct Node {
12     int a, b, c;
13     int pending;
14
15     Node() {}
16     Node(int a) : a(a), b(0), c(0), pending(0) {}
17     Node(int a, int b, int c) : a(a), b(b), c(c), pending(0) {}
18
19     Node change(int n) {
20         n%=3;
21         pending += n;
22         if (n==1) {
23             swap(a, b); swap(a, c);
24         } else if (n==2) {
25             swap(c, b); swap(c, a);
26         }
27         return *this;
28     }
29
30     Node operator +(Node x) {
31         return Node(a+x.a, b+x.b, c+x.c);
32     }
33 };

```

```

34
35 struct Segtree {
36     Node T[MAX];
37     int n;
38
39     Segtree() {
40         clear(1);
41     }
42
43     void clear(int n) {
44         while(n != n&-n)
45             n += n&-n;
46
47         this->n = n;
48
49         build(1, 1, n);
50     }
51
52     void build(int v, int a, int b) {
53         T[v] = Node(b-a+1);
54
55         if (a>=b) return;
56         build(2*v, a, (a+b)/2);
57         build(2*v+1, (a+b)/2+1, b);
58     }
59
60     Node update(int v, int a, int b, int i, int j, int carry, int increment) {
61         T[v].change(carry);
62
63         if (i>b || j<a)
64             return Node(0);
65
66         if (i<=a && b<=j)
67             return T[v].change(increment);
68
69         Node answer =
70             update(v*2, a, (a+b)/2, i, j, T[v].pending, increment) +
71             update(v*2+1, (a+b)/2+1, b, i, j, T[v].pending, increment);
72
73         T[v] = T[v*2] + T[v*2+1];
74
75         return answer;
76     }
77
78     Node update(int i, int j, int inc) {
79         return update(1, 1, n, i, j, 0, inc);
80     }
81
82     Node query(int i, int j) {
83         return update(1, 1, n, i, j, 0, 0);
84     }
85
86 };
87
88 Segtree T;
89
90 int main() {
91     int n, m;
92     while(scanf("%d%d", &n, &m) != EOF) {
93         T.clear(n);
94         for(int i=0; i<m; i++) {
95             char cmd; int a, b;
96             scanf(" %c %d %d", &cmd, &a, &b);
97             if (cmd == 'M') {
98                 T.update(a, b, 1);
99             } else {
100                 Node node = T.query(a, b);
101                 printf("%d %d %d\n", node.a, node.b, node.c);
102             }
103         }
104         printf("\n");
105     }
106 }

```