```cpp
//
//  100 - The 3n + 1 problem.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/2/23.
//

#include <iostream>
using namespace std;

int main() {
    int x, y;
    while (cin >> x >>  y) {
        int m = INT32_MIN;
        for (int i = min(x, y); i <= max(x, y); ++i) {
            int c = 1;
            int n = i;
            while (n != 1) {
                if ((n & 1) != 0)
                    n = 3 * n + 1;
                else
                    n /= 2;
                c++;
            }
            m = max(c, m);
        }
        cout << x << ' ' <<  y << ' ' << m << endl;
    }
}

//
//  The Blocks Problem.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/2/23.
//

#include <iostream>
```

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

struct Node {
    bool atOriginal;
    int prev, next;
    Node(): prev(-1), next(-1), atOriginal(true) {};
};

bool checkValid(int c, vector<Node> &blocks, int b) {
    while (c != -1 && c != b) {
        c = blocks[c].next;
    }
    return c != b;
}

void returnAnyBlock(vector<Node> &blocks, int b) {
    int current = blocks[b].next;
    while (current != -1) {
        blocks[current].atOriginal = true;
        blocks[current].prev = -1;
        int temp = blocks[current].next;
        blocks[current].next = -1;
        current = temp;
    }
}

int main() {
    int n;
    cin >> n;
    vector<Node> blocks(n);
    string firstCommand;
    cin >> firstCommand;
    while (firstCommand != "quit") {
        int a, b;
        string secondCommand;
        cin >> a >> secondCommand >> b;
```

```
        bool isValid = a!=b && checkValid(a, blocks, b) && checkValid(b,
blocks, a);
        if (!isValid) {
            cin >> firstCommand;
            continue;
        }
        if (secondCommand == "onto") {
            returnAnyBlock(blocks, b);
        } else {
            // move to top of b
            while (blocks[b].next != -1) {
                b = blocks[b].next;
            }
        }
        if (firstCommand == "move") {
            returnAnyBlock(blocks, a);
            blocks[a].next = -1;
        }
        // move a off of previous
        if (blocks[a].prev != -1)
            blocks[blocks[a].prev].next = -1;
        // put a on top of b
        blocks[b].next = a;
        blocks[a].prev = b;
        blocks[a].atOriginal = false;
        cin >> firstCommand;
    }
    for (int i=0; i<n; i++) {
        cout << i << ':';
        if (blocks[i].atOriginal) {
            int x = i;
            while (x != -1) {
                cout << ' ' << x;
                x = blocks[x].next;
            }
        }
        cout << endl;
    }
```

```
}

//
//  Ecological Bin Packing.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/2/26.
//

#include <iostream>
using namespace std;

int main() {
    int a[10];
    const string COLORS = " BGC";
    while (cin >> a[1]) {
        for (int i=2; i<=9; ++i) {
            cin >> a[i];
        }
        int minMove = INT32_MAX;
        string r = "";
        for (int i=1; i<=3; ++i) {
            int moveLastTwo = a[3+i] + a[6+i];
            for (int j=1; j<=3; ++j) {
                if (j == i) continue;
                int addMoveFirstThird = moveLastTwo + a[j] + a[6+j];
                for (int k=1; k<=3; ++k) {
                    if (k == i || k == j) continue;
                    int addMoveFirstTwo = addMoveFirstThird + a[k] + a[3+k];
                    if (minMove >= addMoveFirstTwo) {
                        string temp = "";
                        temp += COLORS[i];
                        temp += COLORS[j];
                        temp += COLORS[k];
                        if (minMove > addMoveFirstTwo || r == "" || r > temp)
                            r = temp;
                        minMove = addMoveFirstTwo;
                    }
```

```
            }
        }
    }

    cout << r << ' ' << minMove << endl;
  }
}

//
//  Stacking Boxes.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/2/26.
//

#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

bool compareBox(const vector<int> &l, const vector<int> &r) {
    for (int i=0; i<l.size()-1; i++) {
        if (l[i] == r[i]) continue;
        return l[i] > r[i];
    }
    return l.back() < r.back();
}

bool canFit(const vector<int> &large, const vector<int> &small) {
    for (int i=0; i<large.size()-1; i++) {
        if (large[i] <= small[i])
            return false;
    }
    return true;
}

int main() {
    int n, dim;
```

```
while (cin >> n >> dim) {
    vector<vector<int>> boxes;
    vector<int> len(n, 1);
    vector<int> prev(n, -1);
    for (int i=0; i<n; i++) {
        vector<int> box(dim, 0);
        for (int j=0; j<dim; j++) {
            cin >> box[j];
        }
        sort(box.begin(), box.end(), greater<int>());
        box.push_back(i+1); // box.back() boxId
        boxes.push_back(box);
    }
    sort(boxes.begin(), boxes.end(), compareBox);
    for (int i=0; i<n; i++) {
        for (int j=i+1; j<n; j++) {
            if (canFit(boxes[i], boxes[j]) && len[i]+1 >= len[j]) {
                len[j] = len[i] + 1;
                prev[j] = i;
            }
        }
    }
    int maxLen = 0;
    int maxIdx = 0;
    for (int i=0; i<n; i++) {
        if (len[i] > maxLen) {
            maxLen = len[i];
            maxIdx = i;
        }
    }
    cout << maxLen << endl;
    int t = maxIdx;
    while (prev[t] != -1) {
        cout << boxes[t].back() << ' ';
        t = prev[t];
    }
    cout << boxes[t].back() << endl;
}
```

```
}

//
//  Arbitrage.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/8.
//

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

int main() {
    int n;
    const int MAX_DIM = 20;
    while (cin >> n) {
        double a[MAX_DIM][MAX_DIM][MAX_DIM] = {0}; // for floyd-
warshall
        int path[MAX_DIM][MAX_DIM][MAX_DIM] = {0};
        for (int i=0; i<n; i++) {
            for (int j=0; j<n; j++) {
                path[i][j][0] = i;
                if (i == j) {
                    a[i][j][0] = 1.0;
                    continue;
                }
                cin >> a[i][j][0];
            }
        }
        for (int l=1; l<n; l++) {   // floyd-warshall algorithm
            for (int k=0; k<n; k++) {
                for (int i=0; i<n; i++) {
                    for (int j=0; j<n; j++) {
                        double x = a[i][k][l-1]*a[k][j][0];
                        if (a[i][j][l] < x) {
                            a[i][j][l] = x;
                            path[i][j][l] = k;  // depth
```

```
                    }
                }
            }
        }
    }
    bool found = false;
    for (int l=1; l<n && !found; l++) {
        for (int i=0; i<n; i++) {
            if (a[i][i][l] > 1.01) {
                vector<int> minLength;
                minLength.push_back(path[i][i][l]);
                for (int j=l-1; j>=0; j--) {
                    minLength.push_back(path[i][minLength.back()][j]);
                }
                int temp = minLength.back();
                while (!minLength.empty()) {
                    cout << minLength.back()+1 << ' ';
                    minLength.pop_back();
                }
                cout << temp+1 << endl;
                found = true;
                break;
            }
        }
    }
    if (!found)
        cout << "no arbitrage sequence exists" << endl;
  }

}

//
//  The Skyline Problem.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/5.
//
```

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

int main() {
    int leftCoordinate, height, rightCoordinate;
    const int MAX_COORDINATE = 20000; // unit is 0.5
    int skyline[MAX_COORDINATE] = {0};
    vector<int> r;
    while (cin >> leftCoordinate >> height >> rightCoordinate) {
        for (int i=leftCoordinate; i<=rightCoordinate; i++) {
            if (skyline[2*i] < height) {
                skyline[2*i] = height;
            }
            if (i != rightCoordinate && skyline[2*i+1] < height) {
                skyline[2*i+1] = height;
            }
        }
    }
    int heightTracking = 0;
    for (int i=0; i<=MAX_COORDINATE; ++i) {
        if (heightTracking != skyline[i]) {
            r.push_back(heightTracking < skyline[i] ? i/2 : (i-1)/2);
            heightTracking = skyline[i];
            r.push_back(heightTracking);
        }
    }
    for (int i = 0; i<r.size()-1; i++) {
        cout << r[i] << ' ';
    }
    cout << r.back() << endl;
}

//
//  Fermat vs.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/8.
```

```
//

#include <iostream>
#include <math.h>
using namespace std;

int gcd(int a, int b) {
    return b ? gcd(b, a%b) : a;
}

int main() {
    int n;
    const int MAX_NUM = 1e6;
    while (cin >> n) {
        bool flag[MAX_NUM+1] = {false};
        int count1 = 0;
        int count2 = 0;
        for (int a=1; a<=(int)sqrt(n); a++) {
            for (int b=a+1; a*a+b*b<=n; b++) {
                if (gcd(a, b) == 1 && a%2 != b%2) {
                    count1++;
                    int x = 2*a*b;
                    int y = b*b-a*a;
                    int z = b*b+a*a;
                    for (int k=1; k*z<=n; k++) {
                        flag[k*x] = flag[k*y] = flag[k*z] = true;
                    }
                }
            }
        }
        for (int i=1; i<=n; i++) {
            count2 += !flag[i];
        }
        cout << count1 << ' ' << count2 << endl;
    }
}

//
```

```
//  The Cat in the Hat.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/9.
//

#include <iostream>
#include <math.h>
using namespace std;

int main() {
    unsigned long long h, workers;
    while (cin >> h >> workers) {
        if (workers==0 && h==0)
            break;
        for (int n=1; ; n++) {
            auto cats = 1;
            auto sumHeight = h;
            auto height = h;
            int spawnTime = 0;
            while (height != 1) {
                if (height%(n+1) != 0)
                    break;
                spawnTime++;
                height /= n+1;
                cats += pow(n, spawnTime);
                sumHeight += pow(n, spawnTime)*height;
            }
            if (height == 1 && cats >= workers) {
                cout << cats-workers << ' ' << sumHeight << endl;
                break;
            }
        }
    }
}

//
//  Maximum Sum.cpp
```

```
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/6.
//

#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    const int MAX_DIM = 101;
    int a[MAX_DIM][MAX_DIM];
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n; j++) {
            cin >> a[i][j];
        }
    }
    int r = INT16_MIN;
    int sumRectangle[MAX_DIM][MAX_DIM] = {0};
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n; j++) {
            sumRectangle[i][j] = sumRectangle[i-1][j] + sumRectangle[i][j-1]
+ a[i][j] - sumRectangle[i-1][j-1];
        }
    }
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n; j++) {
            for (int k=0; k<i; k++) {
                for (int m=0; m<j; ++m) {
                    int subRectangle = sumRectangle[i][j] - sumRectangle[k][j] -
sumRectangle[i][m] + sumRectangle[k][m];
                    r = max(r, subRectangle);
                }
            }
        }
    }
    cout << r << endl;
```

```
}

//
//  Meta-Loopless Sorts.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/7.
//

#include <iostream>
#include <vector>
#include <string>
using namespace std;

const int MAX_DIM = 8;
const string VARIABLE_NAMES = "abcdefgh";

void printVariables(int variables[], int numVars) {
    for (int i=0; i<numVars; i++) {
        if (i != 0)
            cout << ',';
        cout << VARIABLE_NAMES[variables[i]];
    }
}

void printDepth(int depth) {
    cout << string(2*depth, ' ');
}

void compare(int depth, int numVars, int variables[]) {
    if (depth == numVars) {
        printDepth(depth);
        cout << "writeln(";
        printVariables(variables, numVars);
        cout << ')' << endl;
        return;
    }
    int childVars[MAX_DIM];
```

```
    copy(variables, variables+numVars, childVars);
    for (int i=depth-1; i>=0; i--) {
        printDepth(depth);
        if (i!=depth-1)
            cout << "else ";
        cout << "if " << VARIABLE_NAMES[childVars[i]] << " < "
        << VARIABLE_NAMES[childVars[i+1]] << " then" << endl;
        compare(depth+1, numVars, childVars);
        swap(childVars[i], childVars[i+1]);
    }
    printDepth(depth);
    cout << "else\n";
    compare(depth+1, numVars, childVars);
}

int main() {
    int m;
    cin >> m;
    while (m--) {
        int numVars;
        cin >> numVars;
        cout << "program sort(input,output);" << endl;
        cout << "var\n";
        int variables[MAX_DIM];
        for (int i=0; i<numVars; i++) {
            variables[i] = i;
        }
        printVariables(variables, numVars);
        cout << " : integer;\n";
        cout << "begin\n";
        cout << "  readln(";
        printVariables(variables, numVars);
        cout << ");\n";
        if (numVars == 1) {
            cout << "  writeln(a)\n";
        } else
            compare(1, numVars, variables);
        cout << "end.\n";
```

```
        if (m>0)
            cout << endl;
    }
}

//
//  History Grading.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/9.
//

#include <iostream>
#include <vector>
#include <map>
using namespace std;

int main() {
    int m;
    int n = 0;
    const int MAX_DIM = 21;
    map<int, int> order;
    int seq[MAX_DIM];
    char c = '\n';
    int id = 0;
    while (cin >> m) {
        char t = getchar();
        if (c == '\n' && t == '\n') {
            n = m;
            order.clear();
            for (int i=1; i<=n; i++) {
                cin >> m;
                order[i] = m;
            }
            continue;
        }
        c = t;
        seq[m] = ++id;
```

```
        if (c == '\n') {
            for (int i=1; i<=n; i++) {
                seq[i] = order[seq[i]];
            }
            int r = 0;
            vector<int> dp(MAX_DIM, 1);
            for (int i=2; i<=n; i++) {
                for (int j=1; j<i; j++) {
                    if (seq[i] > seq[j])
                        dp[i] = max(dp[i], dp[j]+1);
                }
                r = max(r, dp[i]);
            }
            cout << r << endl;
            id = 0;
        }
    }
}

//
//  Tree Summing.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/10.
//

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

int sumNumbers(vector<int> numbers) {
    int s=0;
    for (int i : numbers)
        s += i;
    return s;
}

int main() {
```

```
int n;
while (cin >> n) {
    int parentheses = 0;    // for count parentheses
    vector<int> numbers;
    string t;
    char c;
    bool hasNumber = false;
    int countNotHasNumber = 0;
    bool found = false;
    while (cin >> c) {
        if (isdigit(c) || c == '-')
            t += c;
        else if (t.length()>0) {
            numbers.push_back(stoi(t));
            hasNumber = true;
            countNotHasNumber = 0;
            t = "";
        }
        if (c == '(') {
            parentheses++;
            hasNumber = false;
        }
        if (c == ')') {
            if (!hasNumber)
                countNotHasNumber++;
            else {
                numbers.pop_back();
                countNotHasNumber = 0;
            }
            if (countNotHasNumber == 2) {
                if (n == sumNumbers(numbers)) {
                    found = true;
                }
                countNotHasNumber = 0;
            }
            parentheses--;
            hasNumber = true;   // for pop back numbers
            if (parentheses == 0)   // end of tree
```

```
                break;
            }
        }
        cout << (found ? "yes" : "no") << endl;
    }
}

//
//  Power of Cryptography.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/10.
//

#include <iostream>
#include <math.h>
#include <iomanip>
using namespace std;

int main() {
    double n, p;
    while (cin >> n >> p) {
        cout << fixed << setprecision(0) << pow(p, 1/n) << endl;
    }
}

//
//  Climbing Trees.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/11.
//

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

```
int main() {
    map<string, string> dad;
    string left, right;
    while (cin >> left >> right && left != "no.child") {
        dad[left] = right;
    }
    while (cin >> left >> right) {
        if (dad.count(left) && dad[left] == right)
            cout << "child";
        else if (dad.count(right) && dad[right] == left)
            cout << "parent";
        else if (dad.count(right) && dad.count(left) && dad[right] ==
dad[left])
            cout << "sibling";
        else {
            bool found = false;
            string temp = left;
            int level = 0;
            while (dad.count(temp) && !found) {
                level++;
                temp = dad[temp];
                found = temp == right;
            }
            if (found) {
                for (int i=level; i>2; i--)
                    cout << "great ";
                cout << "grand child";
            }
            if (!found) {
                temp = right;
                level = 0;
                while (dad.count(temp) && !found) {
                    level++;
                    temp = dad[temp];
                    found = temp == left;
                }
                if (found) {
                    for (int i=level; i>2; i--)
```

```
                cout << "great ";
            cout << "grand parent";
        }
    }
    if (!found) {
        vector<string> leftAncestors, rightAncestors;
        temp = left;
        while (dad.count(temp)) {
            temp = dad[temp];
            leftAncestors.push_back(temp);
        }
        temp = right;
        while (dad.count(temp)) {
            temp = dad[temp];
            rightAncestors.push_back(temp);
        }
        int i, j;
        for (i=0; i<leftAncestors.size() && !found; i++) {
            for (j=0; j<rightAncestors.size() && !found; j++) {
                found = leftAncestors[i] == rightAncestors[j];
            }
        }
        if (found) {
            cout << min(i, j)-1 << " cousin";   // for loop still increment
before break
            if (i != j)
                cout << " removed " << abs(i-j);
        }
    }
    if (!found)
        cout << "no relation";
  }
  cout << endl;
 }
}

#include <iostream>
#include <vector>
```

```
using namespace std;

int main () {
    int x, y;
    while (cin >> x >> y) {
        int z[10][100];
        int dp[10][100];
        vector<int> path;
        for (int i = 0; i < x; i++)
            for (int j = 0; j < y; j++)
                cin >> z[i][j];
        for (int i = 0; i < x; i++)
            dp[i][y-1] = z[i][y-1];
        for (int j = y - 2; j >= 0; j--) {
            for (int i = 0; i < x; i++) {
                int a = dp[(i-1 + x) % x][j+1];
                int b = dp[i][j+1];
                int c = dp[(i+1 + x) % x][j+1];
                dp[i][j] = min(a, min(b, c)) + z[i][j];
            }
        }
        int best = dp[0][0];
        int id = 0;
        for (int i = 1; i < x; i++) {
            if (dp[i][0] < best) {
                best = dp[i][0];
                id = i;
            }
        }
        path.push_back(id);
        for (int j = 1; j < y; j++) {
            int temp = dp[id][j-1] - z[id][j-1];
            int tempId = INT32_MAX;
            int p = (id - 1+x) % x;
            int q = (id + 1+x) % x;
            if (dp[p][j] == temp) {
                tempId = min(tempId, p);
            }
```

```
            if (dp[q][j] == temp) {
                tempId = min(tempId, q);
            }
            if (dp[id][j] == temp) {
                tempId = min(tempId, id);
            }
            id = tempId;
            path.push_back(id);
        }
        for (int i=0; i<path.size(); i++) {
            if (i != 0)
                cout << ' ';
            cout << path[i] + 1;
        }
        cout << endl << best << endl;
    }
}

//
//  Mutant Flatworld Explorers.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/12.
//

#include <iostream>
#include <map>
using namespace std;

int main() {
    int n, m;
    cin >> n >> m;
    string orientations = "NESW";
    map<char, int> oriToInt;
    map<int, char> intToOri;
    for (int i=0; i<orientations.length(); i++) {
        oriToInt[orientations[i]] = i;
        intToOri[i] = orientations[i];
```

```
    }
    const int MAX_DIM = 50;
    bool scent[MAX_DIM][MAX_DIM];
    int x, y;
    char orientation;
    string instruction;
    while (cin >> x >> y >> orientation >> instruction) {
        int oriInt = oriToInt[orientation];
        bool isLost = false;
        for (char i : instruction) {
            if (i == 'R')
                oriInt = (oriInt+1)%4;
            else if (i == 'L')
                oriInt = (oriInt-1+4)%4;
            else {
                orientation = intToOri[oriInt];
                int yt = y + (orientation == 'N') - (orientation == 'S');
                int xt = x + (orientation == 'E') - (orientation == 'W');
                if (xt>n || yt>m || xt<0 || yt<0) {
                    if (!scent[x][y]) {
                        isLost = true;
                        scent[x][y] = true;
                        break;
                    }
                } else {
                    x = xt;
                    y = yt;
                }
            }
        }
        cout << x << ' ' << y << ' ' << intToOri[oriInt] << (isLost ? "
LOST\n" : "\n");
    }
}

//
//  Greedy Gift Givers.cpp
//  OnlineJudge
```

```
//
//  Created by Tien Do on 2021/5/13.
//

#include <iostream>
#include <map>
#include <vector>
using namespace std;

int main() {
    string s;
    int n;
    int count = 0;
    while (cin >> n) {
        vector<string> names;
        map<string, int> r;
        string name;
        for (int i=0; i<n; i++) {
            cin >> name;
            names.push_back(name);
            r[name] = 0;
        }
        for (int i=0; i<n; i++) {
            int money, p;
            cin >> name >> money >> p;
            if (p != 0) {
                r[name] -= (money/p)*p;
                for (int i=0; i<p; i++) {
                    cin >> name;
                    r[name] += money/p;
                }
            }
        }
        if (count != 0)
            cout << endl;
        for (string name : names) {
            cout << name << ' ' << r[name] << endl;
        }
```

```
        count++;
    }
}

//
//  Stacks of Flapjacks.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/13.
//

#include <iostream>
#include <algorithm>
using namespace std;

void reverseArray(int a[], int j) {
    for (int i=0; i<=j/2; i++) {
        int temp = a[i];
        a[i] = a[j-i];
        a[j-i] = temp;
    }
}

int main() {
    int n;
    int v[30], vSorted[30];
    int h = 0;
    while (cin >> n) {
        v[h] = n;
        h++;
        if (getchar() != '\n') {
            continue;
        }
        for (int i=0; i<h; i++)
            cout << v[i] << ' ';
        cout << endl;
        copy(v, v+h, vSorted);
        sort(vSorted, vSorted+h);
```

```
        for (int i=h-1; i>0; i--) {
            if (v[i] == vSorted[i])
                continue;
            for (int j=i-1; j>0; j--) {
                if (vSorted[i] == v[j]) {
                    cout << h-j << ' ';
                    reverseArray(v, j);
                    break;
                }
            }
            cout << h-i << ' ';
            reverseArray(v, i);
        }
        cout << "0\n";
        h = 0;
    }
}

//
//  Pipe Fitters.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/14.
//

#include <iostream>
#include <math.h>
using namespace std;

int main() {
    double a, b;
    while (cin >> a >> b) {
        int s = 0;
        double reduceHeight = 1 - 0.5*sqrt(3);
        int numberRows = (int)((a - reduceHeight) / (1 - reduceHeight));
        int b1 = (int)b;
        int sumBySkew = numberRows*b1 - (b-b1 < 0.5) * numberRows/2;
        s = max(s, sumBySkew);
```

```
        numberRows = (int)((b - reduceHeight) / (1 - reduceHeight));
        int a1 = (int)a;
        sumBySkew = numberRows*a1 - (a-a1 < 0.5) * numberRows/2;
        s = max(s, sumBySkew);
        if (a1*b1 >= s)
            cout << a1*b1 << " grid\n";
        else
            cout << s << " skew\n";
    }
}

//
//  Pipe Fitters.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/14.
//

#include <iostream>
#include <math.h>
using namespace std;

int main() {
    double a, b;
    while (cin >> a >> b) {
        int s = 0;
        double reduceHeight = 1 - 0.5*sqrt(3);
        int numberRows = (int)((a - reduceHeight) / (1 - reduceHeight));
        int b1 = (int)b;
        int sumBySkew = numberRows*b1 - (b-b1 < 0.5) * numberRows/2;
        s = max(s, sumBySkew);
        numberRows = (int)((b - reduceHeight) / (1 - reduceHeight));
        int a1 = (int)a;
        sumBySkew = numberRows*a1 - (a-a1 < 0.5) * numberRows/2;
        s = max(s, sumBySkew);
        if (a1*b1 >= s)
            cout << a1*b1 << " grid\n";
        else
```

```
            cout << s << " skew\n";
    }
}

//
//  Searching Quickly.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/14.
//

#include <iostream>
#include <sstream>
#include <map>
#include <set>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    string s;
    vector<string> keywords;
    set<string> ignore;
    map<string, vector<string> > m;
    while (cin >> s && s != "::") {
        ignore.insert(s);
    }
    while (getline(cin, s)) {
        string t;
        for (char i : s) {
            t += tolower(i);
        }
        stringstream ss(t);
        vector<string> words;
        while (ss >> s) {
            words.push_back(s);
        }
        for (int i=0; i<words.size(); i++) {
```

```
            string word = words[i];
            if (ignore.count(word))
                continue;
            keywords.push_back(word);
            t = "";
            for (int j=0; j<words.size(); j++) {
                if (j!= 0)
                    t += " ";
                if (j != i) {
                    t += words[j];
                } else {
                    s = "";
                    for (char k : word) {
                        s += toupper(k);
                    }
                    t += s;
                }
            }
            m[word].push_back(t);
        }
    }
    sort(keywords.begin(), keywords.end());
    for (int i=0; i<keywords.size(); i++) {
        string word = keywords[i];
        if (i>0 && keywords[i-1] == word)
            continue;
        for (string title : m[word]) {
            cout << title << endl;
        }
    }
}

//
//  Following Orders.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/15.
//
```

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

map<char, bool> visited;
map<char, vector<char> > biggers;
vector<char> chars;

bool isValid(char c) {
    for (char bigger : biggers[c]) {
        if (visited[bigger])
            return false;
    }
    return true;
}

void generateCombination(string comb) { // permutation
    if (comb.length() == chars.size())
        cout << comb << endl;
    for (char c : chars) {
        if (visited[c])
            continue;
        visited[c] = true;
        if (isValid(c))
            generateCombination(comb + c);
        visited[c] = false;
    }
}

int main() {
    char a, b;
    string s;
    int i=0;
```

```
    while (getline(cin, s)) {
        if (i!=0)
            cout << endl;
        i++;
        visited.clear();
        biggers.clear();
        chars.clear();
        stringstream ss(s);
        while (ss >> a) {
            chars.push_back(a);
        }
        sort(chars.begin(), chars.end());
        getline(cin, s);
        ss.clear();
        ss.str(s);
        while (ss >> a >> b) {
            biggers[a].push_back(b);
        }
        generateCombination("");
    }
}

//
//  Numbering Paths.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/16.
//

#include <iostream>
using namespace std;

int main() {
    int m;
    int city = 0;
    while (cin >> m) {
        int n = 0;
        const int DIM = 30;
```

```
        int dp[DIM][DIM] = {0}; // must be initialized to 0
        int a, b;
        for (int i=0; i<m; i++) {
            cin >> a >> b;
            dp[a][b] = 1;
            n = max(n, a);
            n = max(n, b);
        }
        n++;
        for (int k=0; k<n; k++) {
            for (int i=0; i<n; i++) {
                for (int j=0; j<n; j++) {
                    dp[i][j] += dp[i][k] * dp[k][j];
                }
            }
        }
        for (int k=0; k<n; k++) {
            if (dp[k][k])
                for (int i=0; i<n; i++) {
                    for (int j=0; j<n; j++) {
                        if (dp[i][k] && dp[k][j])
                            dp[i][j] = -1;
                    }
                }
        }
        cout << "matrix for city " << city++ << endl;
        for (int i=0; i<n; i++) {
            for (int j=0; j<n; j++) {
                if (j)
                    cout << ' ';
                cout << dp[i][j];
            }
            cout << endl;
        }
    }
}

//
```

```
//  The Errant Physicist.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/16.
//

#include <iostream>
#include <vector>
#include <string>
using namespace std;

string r1, r2;

struct Term {
    int coef, xExp, yExp;
};

void getTerm(string s, int &n, Term* terms) {
    string term;
    for (int i=0; i<s.length(); i++) {
        char c = s[i];
        bool isNewTerm = c == '+' || c == '-';
        if ((isNewTerm && i) || i == s.length()-1) {
            string subTerm = "";
            vector<string> subTerms;
            if (i == s.length() -1)
                term += c;
            for (char j : term) {
                if (j == 'y' || j == 'x') {
                    subTerms.push_back(subTerm);
                    subTerm = "";
                }
                subTerm += j;
            }
            subTerms.push_back(subTerm);
            for (string k : subTerms) {
                char firstChar = k[0];
                if (firstChar == 'x') {
```

```
                k = k.substr(1);
                terms[n].xExp = k == "" ? 1 : stoi(k);
            } else if (firstChar == 'y') {
                k = k.substr(1);
                terms[n].yExp = k == "" ? 1 : stoi(k);
            } else {
                if (k == "-")
                    terms[n].coef = -1;
                else if (k == "+" || k == "")
                    terms[n].coef = 1;
                else
                    terms[n].coef = stoi(k);
            }
        }
        term = "";
    }
    term += c;
    n += (isNewTerm && i) || i == s.length()-1;
  }
}

void buildR2(string is) {
  r2 += is;
  r1 += string(is.length(), ' ');
}

void buildR1(int i) {
  if (i != 1) {
    string is = to_string(i);
    r1 += is;
    r2 += string(is.length(), ' ');
  }
}

int main() {
  string s;
  while (cin >> s && s != "#") {
    const int DIM = 80;
```

```
Term terms1[DIM] = {};
int n=0;
getTerm(s, n, terms1);
cin >> s;
Term terms2[DIM] = {};
int m = 0;
getTerm(s, m, terms2);
const int EX_DIM = 200;
int coefficients[EX_DIM][EX_DIM] = {0};
int maxX = 0;
int maxY = 0;
for (int i=0; i<n; i++) {
    for (int j=0; j<m; j++) {
        int x = terms1[i].xExp + terms2[j].xExp;
        int y = terms1[i].yExp + terms2[j].yExp;
        maxX = max(maxX, x);
        maxY = max(maxY, y);
        coefficients[x][y] += terms1[i].coef * terms2[j].coef;
    }
}
r1 = "";
r2 = "";
for (int i=maxX; i>=0; i--) {
    for (int j=0; j<=maxY; j++) {
        int coefficient = coefficients[i][j];
        if (coefficient) {
            if (r1 != "")
                buildR2(coefficient > 0 ? " + " : " - ");
            else if (coefficient < 0)
                buildR2("-");
            int coefficientAbs = abs(coefficient);
            if (i==0 && j==0 && coefficientAbs == 1)
                buildR2("1");
            else if (coefficientAbs != 1)
                buildR2(to_string(coefficientAbs));
            if (i>0) {
                buildR2("x");
                buildR1(i);
```

```
                }
                if (j>0) {
                    buildR2("y");
                    buildR1(j);
                }
            }
        }
        cout << r1 << endl << r2 << endl;
    }
}

//
//  127 - "Accordian" Patience.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/17.
//

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

int main() {
    string s;
    while (cin >> s && s != "#") {
        vector<string> pile(1, s);
        vector<vector<string> > piles(1, pile);
        for (int i=0; i<51; i++) {
            cin >> s;
            pile.clear();
            pile.push_back(s);
            piles.push_back(pile);
            bool makeMove = true;
            while (makeMove) {
                makeMove = false;
                for (int j=1; j<piles.size(); j++) {
                    string top = piles[j].back();
```

```
                if (j>2 && (top[0] == piles[j-3].back()[0] || top[1] ==
piles[j-3].back()[1])) {
                    piles[j-3].push_back(top);
                    piles[j].pop_back();
                    makeMove = true;
                    break;
                } else if (top[0] == piles[j-1].back()[0] || top[1] ==
piles[j-1].back()[1]) {
                    piles[j-1].push_back(top);
                    piles[j].pop_back();
                    makeMove = true;
                    break;
                }
            }
            if (makeMove) {
                for (int j=0; j<piles.size(); j++) {
                    if (piles[j].empty()) {
                        piles.erase(piles.begin()+j);
                    }
                }
            }
        }
        cout << piles.size() << (piles.size() == 1 ? " pile" : " piles") << "
remaining:";
        for (vector<string> s : piles) {
            cout << ' ' << s.size();
        }
        cout << endl;
    }
}

//
//  128 - Software CRC.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/17.
//
```

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

int main() {
    string s;
    const int g = 34943;
    while (getline(cin, s) && s != "#") {
        if (s.length() == 0) {
            cout << "00 00" << endl;
            continue;
        }
        long n = s[0];
        for (int i=1; i<s.size(); i++) {
            n %= g;
            n <<= 8;
            n += s[i];
        }
        n <<= 16;
        n = g - (n%g);
        string hexNum="";
        char hex[]={'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
        while (n>0) {
            int r = n%16;
            hexNum = hex[r] + hexNum;
            n /= 16;
        }
        hexNum = string(4-hexNum.length(), '0') + hexNum;
        cout << hexNum[0] << hexNum[1] << ' ' << hexNum[2] <<
hexNum[3] << endl;
    }
}

//
//  129 - Krypton Factor.cpp
//  OnlineJudge
//
//  Created by Tien Do on 2021/5/19.
```

```
//

#include <iostream>
using namespace std;

char a[81];
int n, l, len;

bool check(int cur) {
    for (int i=1; i*2 <= cur+1; i++) {  // size is cur+1
        bool same = true;
        int p = cur;
        int k = cur-i;
        for (int j=0; j<i; j++) {
            if (a[p--] != a[k--])   // compare start from current
                same = false;
        }
        if (same) return false;
    }
    return true;
}

void dfs(int cur) {
    if (cur && --n == 0) {  // Only descrement after second recursion
        len = cur;
    }
    for (char i='A'; i<'A'+l && n; i++) {   // Halt all loop when n = 0
        a[cur] = i;
        if (!check(cur))
            continue;
        dfs(cur+1);
    }
}

int main() {
    while (cin >> n >> l && n != 0) {
        dfs(0);
        for (int i=0; i<len; i++) {
```

```
            if (i && i%64==0) cout << endl;
            else if (i && i%4==0) cout << ' ';
            cout << a[i];
        }
        cout << endl << len << endl;
    }
}
```