1067. Disk Tree
Для решения задачи создаём структуру директории с именем и поддиректориями на основе std:map.
Считываем новую строку и разделяем её (разделитель - обратный слэш "\") и создаём, если не создано, необходимое "дерево" каталогов (все каталоги, которые есть в пути к файлу). После добавления всех файлов и каталогов, выводим дерево файлов при помощи итератора.

1494. Монобильярд
Создадим двумерный массив. Значения в первом столбце будут соответствовать введённым данным, а значения второго столбца будем использовать как вспомогательные для решения задачи (будем эмулировать стэк). Будем считать, что Чичиков невиновен, пока не докажем обратное.

Сначала мы кладём на вершину стека шар 1. Затем сравниваем вершину стека и каждый элемент. Если они равны, то убираем элемент со стека и идем к следующему элементу, в противном случае закидываем шар со следующим номером в стек (проверяя, не закончились ли у нас шары, естественно). Если у нас закончились шары и при этом вершина стека и очередной шар не совпадают, это значит, что Чичиков не мог закатить шары в правильном порядке.

1628. Белые полосы

Отмечаем (помещаем в массив cell[]) пары (x,y) невезения, границ по столбцам и по строкам. Например, для ввода второго теста:
5 1 2
2 1
3 1

O - пустая, X - отмеченная

	0	1	2
0		X	
1	X	O	X
2	X	X	X
3	X	X	X
4	X	O	X
5	X	O	X
6		X	

Сортируем полученный массив в порядке возрастания (по строкам, координате x) и ищем два элемента одной строки, у которых разница между координатами y больше либо равна 2. Почему 2? Потому что, учитывая, что мы отметили границы, разница 2 соответствует полосе длиной 1 (в примере сверху 2-0=2 для строки с индексом 1, например). Если она равна 2, мы сохраняем эту клетку на случай, если при проходе через неё во время проверки по столбцам она станет частью вертикальной полосы счастья
//чего-то большего (условие максимальности по включению).

Сортируем полученный массив в порядке возрастания (по столбцам, координате y) и ищем идентичные элементы для столбцов: два элемента одного столбца, у которых разница между координатами x больше либо равна 2. Отличие: если мы нашли одинокую клетку, то мы проверяем, не находили ли мы её ранее. Да? - Это полоса 1x1, лучше чем ничего, так что на одну светлую полосу стало больше. Нет? Значит она часть чего-то большего, что мы уже посчитали. Ну а любая полоска длиной больше 2 считается полосой счастья и процветания и также увеличивает итоговое количество.

----------1067----------

```
#include <iostream>
#include <map>
#include <sstream>
#include <string>

using namespace std;

struct dir
{
    std::map<std::string, dir*> subs;
} dirs[50001];
int p = 1;

dir* addDir(dir* dir, std::string str)
{
    auto& d = dir->subs[str];
    if(!d)
        d = &dirs[p++];
    return d;
}

void print(dir* dir, int depth = 0)
{
    for(auto s : dir->subs)
    {
        for(int i = 0; i < depth; i++)
            cout << " ";
        cout << s.first << "\n";
        print(s.second, depth+1);
    }
}

int main()
{
    int N;
    cin >> N;
    for(int i = 0; i < N; i++)
    {
        string str, dirstr;
        cin >> str;
        stringstream ss(str);
        dir* dir = &dirs[0];
        while (getline(ss, dirstr, '\\'))
            dir = addDir(dir, dirstr);
    }
    print(&dirs[0]);
}
```

```
--------------------------------------1628--------------------------------------
#include <cstdio>
#include <algorithm>
#include <map>
#include <set>
#include <iostream>

using namespace std;

typedef pair<int, int> Pair;

bool comp1(Pair a, Pair b){
    if(a.first!=b.first) return a.first<b.first;
    return a.second<b.second;
}

bool comp2(Pair a, Pair b){
    if(a.second!=b.second) return a.second<b.second;
    return a.first<b.first;
}

Pair cell[180000];

int main(){
    int N,M,K;
    cin >> N >> M >> K;
    int x, y;

    for(int i = 0; i<K; ++i){
        cin >> x >> y;
        cell[i] = make_pair(x,y);
    }

    for(int i = 1; i<=M; ++i){
        cell[K++] = make_pair(0,i);
        cell[K++] = make_pair(N+1,i);
    }

    for(int i = 1; i<=N; ++i){
        cell[K++] = make_pair(i,0);
        cell[K++] = make_pair(i,M+1);
    }

    int ans = 0;
    set<Pair> S;

    sort(cell,cell+K,comp1);

    for(int i = 0; i+1<K; ++i){
        int diff = cell[i+1].second-cell[i].second;

        if(cell[i].first==cell[i+1].first && diff>=2){
            if(diff==2) S.insert(make_pair(cell[i].first,cell[i].second+1));
            else ++ans;
        }
    }

    sort(cell,cell+K,comp2);

    for(int i = 0;i+1<K;++i){
        int diff = cell[i+1].first-cell[i].first;

        if(cell[i].second==cell[i+1].second && diff>=2){
            if(diff!=2 || S.find(make_pair(cell[i].first+1,cell[i].second))!
```

```
=S.end()){
                    ++ans;
            }
        }
    }

    cout << ans << "\n";

    return 0;
}
```

----------1494----------

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

int main()
{
    int n, n_st=0;
    int st[100000][2];
    cin >> n;
    int number;
    cin >> number;
    st[0][0]=st[0][1]=number;
    for (int i = 0; i < n-1; i++)
    {
        cin >> number;
        if(number==st[n_st][0]-1)
        {
            st[n_st][0]=number;
            if(n_st>0 && st[n_st][0]==st[n_st-1][1]+1)
            {
                st[n_st-1][1]=st[n_st][1];
                n_st--;
            }
        }
        else
        if(number==st[n_st][1]+1)
            st[n_st][1]=number;
        else
        {
            n_st++;
            st[n_st][0]=st[n_st][1]=number;
        }
    }

    (n_st==0 && st[0][0]==1 && st[0][1]==n) ? (cout << "Not a proof\n") : ( cout
<< "Cheater\n");
    return 0;
}
```