Contenido

[Following Orders 5](#_Toc11159810)

[Dollars 6](#_Toc11159811)

[Anagram checker 7](#_Toc11159812)

[Stacking Boxes 9](#_Toc11159813)

[Ecological Bin Packing 10](#_Toc11159814)

[Bandwidth 11](#_Toc11159815)

[The Dole Queue 12](#_Toc11159816)

[Roman Roulette 13](#_Toc11159817)

[Numbering Paths 14](#_Toc11159818)

[The Postal Worker Rings Once 15](#_Toc11159819)

[Trees on the level 17](#_Toc11159820)

[Mutant Flatworld Explorers 18](#_Toc11159821)

[Stacks of Flapjacks 19](#_Toc11159822)

[Arbitrage 20](#_Toc11159823)

[Unidirectional TSP 22](#_Toc11159824)

[Power Crisis 23](#_Toc11159825)

[Traffic Lights 24](#_Toc11159826)

[Beggar My Neighbour 25](#_Toc11159827)

[String Computer 26](#_Toc11159828)

[The Sultan's Successors 27](#_Toc11159829)

[Theseus and the Minotaur 28](#_Toc11159830)

[Clock Patience 29](#_Toc11159831)

[Eeny Meeny 30](#_Toc11159832)

[Laser Lines 30](#_Toc11159833)

[Trip Routing 32](#_Toc11159834)

[Anagram 34](#_Toc11159835)

[Rare Order 35](#_Toc11159836)

[FireTruck 37](#_Toc11159837)

[Getting in Line 38](#_Toc11159838)

[Othello 39](#_Toc11159839)

[Correct Move 41](#_Toc11159840)

[Puzzle 42](#_Toc11159841)

[Testing the CATCHER 44](#_Toc11159842)

[Crossword Answers 45](#_Toc11159843)

[Calling Circles 46](#_Toc11159844)

[Cube Painting 48](#_Toc11159845)

[Quicksome Squares 49](#_Toc11159846)

[Software Allocation 49](#_Toc11159847)

[Count Cantor 51](#_Toc11159848)

[Simply Syntax 52](#_Toc11159849)

[Chess 53](#_Toc11159850)

[Vertex 53](#_Toc11159851)

[The House of Santa Claus 54](#_Toc11159852)bfs

[Divisors 55](#_Toc11159853)

[Safebreaker 56](#_Toc11159854)

[Quadtrees 57](#_Toc11159855)

[Train Swapping 59](#_Toc11159856)

[John's Trip 59](#_Toc11159857)

[Joseph 61](#_Toc11159858)

[Packets 61](#_Toc11159859)

[Robot 62](#_Toc11159860)

[Network 64](#_Toc11159861)

[The New Villa 65](#_Toc11159862)

[Factorial Frequencies 67](#_Toc11159863)

[Mapping the Swaps 73](#_Toc11159864)

[Identifying Concurrent Events 74](#_Toc11159865)

[A Node Too Far 76](#_Toc11159866)

[Packets 77](#_Toc11159867)

[What Base Is This? 79](#_Toc11159868)

[Roman Digititis 80](#_Toc11159869)

[Pseudo-Random Numbers 81](#_Toc11159870)

[Seasonal War 82](#_Toc11159871)

[Pesly Palindromes 84](#_Toc11159872)

[The Bases are Loaded 84](#_Toc11159873)

[Let Me Count The Ways 86](#_Toc11159874)

[Combinations 86](#_Toc11159875)

[WhatFix Notation 87](#_Toc11159876)

[Cowculation 87](#_Toc11159877)

[Intersecting Lines 89](#_Toc11159878)

[Perfection 91](#_Toc11159879)

[Shipping Routes 91](#_Toc11159880)

[Perfect Cubes 93](#_Toc11159881)

[Basically Speaking 93](#_Toc11159882)

[Mapmaker 94](#_Toc11159883)

[Equation Elation 96](#_Toc11159884)

[Palindromes 99](#_Toc11159885)

[M\*A\*S\*H 100](#_Toc11159886)

[Prime Cuts 101](#_Toc11159887)

[Uniform Generator 102](#_Toc11159888)

[Station Balance 102](#_Toc11159889)

[Pi 103](#_Toc11159890)

[Machined Surfaces 103](#_Toc11159891)

[LED Test 104](#_Toc11159892)

[Word Index 105](#_Toc11159893)

[Word-Search Wonder 107](#_Toc11159894)

[MPI Maelstrom 108](#_Toc11159895)

[Integer Inquiry 109](#_Toc11159896)

[Word Transformation 114](#_Toc11159897)

[Bank (Not Quite O.C.R.) 116](#_Toc11159898)

[Block Voting 118](#_Toc11159899)

[Arbirage II 118](#_Toc11159900)

[The Tower of Babylon 120](#_Toc11159901)

[The Circumference of the Circle 120](#_Toc11159902)

[Knight Moves 122](#_Toc11159903)

[Eeny Meeny Moo 123](#_Toc11159904)

[Lotto 124](#_Toc11159905)

[Humble Numbers 125](#_Toc11159906)

[Kibbles `n' Bits `n' Bits `n' Bits 125](#_Toc11159907)

[Little Black Book 127](#_Toc11159908)

[Project Scheduling 128](#_Toc11159909)

[Anagrams 130](#_Toc11159910)

[Periodic Strings 131](#_Toc11159911)

[Graph Connectivity 131](#_Toc11159912)

[Overlapping Rectangles 132](#_Toc11159913)

[Bridge Hand Evaluator 133](#_Toc11159914)

[Mirror, Mirror 134](#_Toc11159915)

[Magic Numbers 136](#_Toc11159916)

[Heads / Tails Probability 136](#_Toc11159917)

[Point in Figures Rectangles 137](#_Toc11159918)

[Point in Figures Rectangles and Circles 138](#_Toc11159919)

[Point in Figures Rectangles,Circles and Triangles 140](#_Toc11159920)

[Permutation Arrays 142](#_Toc11159921)

[The Department of Redundancy Department 143](#_Toc11159922)

[Boggle Blitz 144](#_Toc11159923)

[Fibonacci Freeze 145](#_Toc11159924)

[Simply Subsets 147](#_Toc11159925)

[Polly the Polynomial 148](#_Toc11159926)

[What's The Frequency, Kenneth? 148](#_Toc11159927)

[Black Box 149](#_Toc11159928)

[Jill Rides Again 150](#_Toc11159929)

[Rails 151](#_Toc11159930)

[Prime Ring Problem 151](#_Toc11159931)

[String Distance and Transform Process 152](#_Toc11159932)

[Binomial Showdown 154](#_Toc11159933)

[Compromise 154](#_Toc11159934)

[Dungeon Master 156](#_Toc11159935)

[Tree Recovery 157](#_Toc11159936)

[The Settlers of Catan 158](#_Toc11159937)

[Team Queue 159](#_Toc11159938)

[Error Correction 160](#_Toc11159939)

[Goldbach's Conjecture 161](#_Toc11159940)

[Heavy Cargo 163](#_Toc11159941)

[Head 164](#_Toc11159942)

[Image Recognizer 165](#_Toc11159943)

[Tree 167](#_Toc11159944)

[Multiplying by Rotation 168](#_Toc11159945)

[Nesting a Bunch of Brackets 168](#_Toc11159946)

[Filling the Gaps 170](#_Toc11159947)

[Bridge Hands 172](#_Toc11159948)

[Amazing 173](#_Toc11159949)

[Wormholes 175](#_Toc11159950)

[Dividing Coins 176](#_Toc11159951)

[Crimewave 177](#_Toc11159952)

[Risk 179](#_Toc11159953)

[Just the facts 180](#_Toc11159954)

[Jugs 180](#_Toc11159955)

[Oil Deposits 181](#_Toc11159956)

[Sum It Up 182](#_Toc11159957)

[Skew Binary 184](#_Toc11159958)

[Haiku Review 184](#_Toc11159959)

[ClockHands 185](#_Toc11159960)

[Prime Factors 185](#_Toc11159961)

[Bowling 186](#_Toc11159962)

[Always on the run 187](#_Toc11159963)

[Box of bricks 188](#_Toc11159964)

[Bundling Newspapers 189](#_Toc11159965)

[The Forrest for the Trees 190](#_Toc11159966)

[The Path 191](#_Toc11159967)

[Stacking Boxes 193](#_Toc11159968)

[Arbitrage 194](#_Toc11159969)

[Unidirectional TSP 195](#_Toc11159970)

[The Postal Worker Rings Once 197](#_Toc11159971)

[Mutant Flatworld Explorers 198](#_Toc11159972)

[Stacks of Flapjacks 200](#_Toc11159973)

[Trees on the level 201](#_Toc11159974)

[Following Orders 202](#_Toc11159975)

[Numbering Paths 203](#_Toc11159976)

[Roman Roulette 204](#_Toc11159977)

[The Dole Queue 205](#_Toc11159978)

[Bandwidth 206](#_Toc11159979)

[Dollars 208](#_Toc11159980)

[Anagram checker 208](#_Toc11159981)

[Counterfeit Dollar 211](#_Toc11159982)

[Street Directions 212](#_Toc11159983)

[Coconuts, Revisited 213](#_Toc11159984)

[Secret Research 214](#_Toc11159985)

[500! 214](#_Toc11159986)

[CD 217](#_Toc11159987)

[Ecosystem 218](#_Toc11159988)

[Passwords 219](#_Toc11159989)

[Self Numbers 221](#_Toc11159990)

[Word Amalgamation 221](#_Toc11159991)

[Inmediate Decodability 222](#_Toc11159992)

# Following Orders

\*\*\*ID: 124

\*\*\*Tipo: Graph Theory, Topological Sort, Backtracking

#include <bits/stdc++.h>

#include <sstream>

using namespace std;

#define MAX 35

int used[ MAX ];

///reglas representadas con entero a < b -> [ 0 ][ 1 ] = true

int contraint[ MAX ][ MAX ];

char let[ MAX ];

int idx;

char output[ MAX + 5 ];

bool check( int len , int in ){

//si x ejemplo tengo a b f g

//contrait a <b b < f

//primero entra "a" y se compara con "b" "f" y "g"

//contradiccion al contrait es q a > b x ello output va en indice j

for( int i = 0 ; i < len ; ++i ){

if( contraint[ in ][ output[ i ] - 'a' ] ) return false;

}

return true;

}

void dfs( int len ){

if( len == idx ){

output[ len ] ='\0';

printf("%s\n", output );

return;

}

for( int i = 0 ; i < idx ; ++i ){

if( !used[ i ] && check( len , let[ i ] - 'a' ) ){

output[ len ] = let[ i ];

used[ i ] = 1;

dfs( len + 1 );

used[ i ] = 0;

}

}

}

int main(){

char line[ MAX + 5 ];

char a ,b;

int l;

bool bb = false;

while( gets( line ) ){

if( bb ) putchar( '\n' );

bb = true;

idx = 0;

for( int i = 0 ; line[ i ] ; ++i ){

if( isalpha( line[ i ] ) ) {

let[ idx ] = line[ i ];

used[ idx++ ] = 0;

}

}

//se ordena para sacar lexicograficamente

sort( let, let + idx );

memset( contraint , 0 , sizeof( contraint ) );

gets( line );

strcat( line, " ");

stringstream ss( line );

while( ss>>a>>b ) contraint[ a - 'a' ][ b - 'a' ] = 1;

dfs( 0 );

}

return 0;

}

# Dollars

\*\*\*ID: 147

\*\*\*Tipo: DP, Coin Change

#include <bits/stdc++.h>

using namespace std;

#define MAX 6005

int a[ 11 ] = { 1, 2 , 4 , 10 , 20 , 40 , 100 , 200 , 400 , 1000 , 2000 };

long long dp[ MAX ];

void CoinChange(){

dp[ 0 ] = 1;

for( int i = 0 ; i < 11 ; ++i ){

for( int j = a[ i ] ; j < MAX ; ++j ){

dp[ j ] = dp[ j ] + dp[ j - a[ i ] ];

}

}

}

int main(){

double n;

CoinChange();

while( scanf("%lf" , &n ) && n != 0 ){

printf("%6.2lf%17lld\n" , n , dp[ (int)( n \* 20 ) ] );

}

return 0;

}

# Anagram checker

\*\*\*ID: 148

\*\*\*Tipo: Backtracking, strings

#include <bits/stdc++.h>

using namespace std;

#define MAX 3015

char dictionary[ MAX ][ 42 ];

int cantidad[ MAX ][ 27 ];

int a[ 27 ];

char paragraph[ 42 ];

char resp[ MAX ][ 42 ];

bool seen[ MAX ];

int len , len\_anagram;

char anagram[ 25 ][ 22 ];

void solve( int word , int lon ){

int i;

for( i = 0 ; i < 26 ; ++i ){

if( a[ i ] )break;

}

if( i == 26 ){

int cnt = 0;

for( int k = 0 ; k < lon ; ++k ){

for( int i = 0 ; i < len\_anagram ; ++i ){

if( !strcmp( anagram[ i ] , resp[ k ] ) ){

cnt++;

}

}

}

//si mi respuesta contiene las letras en el mismo orden que la original no va

if( cnt == len\_anagram )return;

printf("%s=" , paragraph );

for( int j = 0 ; j < lon ; ++j ){

printf(" %s" , resp[ j ] );

}

printf("\n");

return;

}

if( word >= len ||lon >= len ){

return;

}

for( int i = word ; i < len ; ++i ){

int j;

for( j = 0 ; j < 26 ; ++j ){

if( a[ j ] >= cantidad[ i ][ j ] )continue;

else break;

}

//si tiene todos los caracteres

if( j == 26 ){

for( j = 0 ; j < 26 ; ++j ){

a[ j ] -= cantidad[ i ][ j ];

}

seen[ i ] = true;

strcpy( resp[ lon ] , dictionary[ i ] );

solve( i + 1 , lon + 1 );

//backtrack

seen[ i ] = false;

for( j = 0 ; j < 26 ; ++j ){

a[ j ] += cantidad[ i ][ j ];

}

}

}

}

int main(){

int lp = 0;

len = 0;

memset( cantidad , 0 , sizeof( cantidad ) );

while( gets( dictionary[ len ] ) && dictionary[ len ][ 0 ] != '#'){

int l = strlen( dictionary[ len ] );

for( int i = 0 ; i < l ; ++i ){

if( 'A' <= dictionary[ len ][ i ] && dictionary[ len ][ i ] <= 'Z' )

cantidad[ len ][ dictionary[ len ][ i ] - 'A' ]++;

}

len++;

}

int aux;

while( gets( paragraph ) , strcmp( paragraph , "#") ){

lp = strlen( paragraph );

len\_anagram = 0;

aux = 0;

memset( seen , 0 , sizeof( seen ) );

bool enter = false;

memset( a , 0 , sizeof( a ) );

paragraph[ lp ] = ' ';

paragraph[ lp + 1 ] = '\0';

for( int i = 0 ; i <= lp ; ++i ){

if( 'A' <= paragraph[ i ] && paragraph[ i ] <= 'Z' ){

anagram[ len\_anagram ][ aux++ ] = paragraph[ i ];

anagram[ len\_anagram][aux] = '\0';

a[ paragraph[ i ] - 'A' ]++;

enter = true;

}

else{

if( enter )len\_anagram++;

enter = false;

aux = 0;

}

}

solve( 0 , 0 );

}

return 0;

}

# Stacking Boxes

\*\*\*ID: 103

\*\*\*Tipo: Sorting, LIS

#include <bits/stdc++.h>

using namespace std;

#define mp make\_pair

#define pb push\_back

#define pvi pair< vector< int > , int >

vector< pvi > v;

vector< int > :: iterator it;

int k;

int dp[ 35 ] , prev[ 35 ];

bool entra( int x , int y ){

for( int i = 0 ; i < k ; ++i )

if( v[ x ].first[ i ] >= v[ y ].first[ i ] ) return false;

return true;

}

void print( int idx , int len ){

if( prev[ idx ] == -1 || len <= 0 ){ printf("%d" , v[ idx ].second );return; }

print( prev[ idx ] , len - 1 );

printf(" %d" , v[ idx ].second );

}

int main(){

int n ,x;

while( scanf("%d %d" , &n , &k ) != EOF ){

v.clear();

vector< int > aux;

for( int i = 0 ; i < n ; ++i ){

aux.clear();

for( int j = 0 ; j < k ; ++j ){

scanf("%d" , &x );

aux.push\_back( x );

}

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

v.pb( mp( aux , i + 1 ) );

dp[ i ] = 1;

prev[ i ] = -1;

}

stable\_sort( v.begin() , v.end() );

for( int i = 0 ; i < n ; ++i ){

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

if( entra( i , j ) ){ //si todos los elementos de la caja i entran en la caja j

if( dp[ j ] < dp[ i ] + 1 ){

dp[ j ] = dp[ i ] + 1;

prev[ j ] = i;

}

}

}

}

int maxi = 0, idx = 0;

for( int i = 0 ; i < n ; ++i ){

if( maxi <= dp[ i ] ){

maxi = dp[ i ];

idx = i;

}

}

printf("%d\n" , maxi );

print( idx , maxi );

printf("\n");

}

return 0;

}

# Ecological Bin Packing

\*\*\*ID: 102

\*\*\*Tipo: Ad hoc, Brute Force

#include <bits/stdc++.h>

using namespace std;

#define MAX 5

#define BLUE 0

#define GREEN 1

#define CLEAR 2

int color[ MAX ][ MAX ];

char s[ 4 ] = "BGC";

int main(){

while( scanf("%d" , &color[0][0] ) != EOF ){

for( int i = 1 ; i < 3 && scanf("%d" , &color[0][i]); ++i );

for( int i = 0 ; i < 3 && scanf("%d" , &color[1][i]); ++i );

for( int i = 0 ; i < 3 && scanf("%d" , &color[2][i]); ++i );

int a[] = { 0 , 1 , 2 };

int mini = 1<<30;

string res = "";

do{

int ans = 0;

for( int i = 0 ; i < 3 ; ++i ){

for( int j = 0 ; j < 3 ; ++j ){

if( a[i] != j ){

ans += color[ i ][ j ];

}

}

}

if( ans < mini ){

mini = ans;

res = "";

for( int i = 0 ; i < 3 ; ++i )

res += s[ a[i]];

}else if( ans == mini ){

string aux = "";

for( int i = 0 ; i < 3 ; ++i )

aux += s[a[i]];

if( aux < res )

res = aux;

}

}while( next\_permutation( a , a + 3 ));

printf("%s %d\n" , res.c\_str() , mini );

}

return 0;

}

# Bandwidth

\*\*\*ID: 140

\*\*\*Tipo: Backtracking

#include <bits/stdc++.h>

using namespace std;

#define MAX 27

vector<int> ady[ MAX ];

bool seen[ MAX ], visited[ MAX ], connected[ MAX ][ MAX ];

int pos[ MAX ];

int len, mini;

string resp;

void dfs( int x , string s ){

if( s.length() == len ){

for( int i = 0 ; i < s.length() ; ++i ){

pos[ s[ i ] - 'A' ] = i + 1;

}

int maxi = 0;

for( int i = 0 ; i < MAX ; ++i ){

if( visited[ i ] ){

for( int j = 0 ; j < ady[ i ].size() ; ++j ){

if( !connected[ i ][ ady[ i ][ j ] ] ) continue;

int aux = abs( pos[ ady[ i ][ j ] ] -pos[ i ] );

if( maxi < aux ){

maxi = aux;

}

}

}

}

if( mini > maxi ){

resp = s;

mini = maxi;

}

}

seen[ x ] = 1;

for( int i = 0 ; i < ady[ x ].size() ; ++i ){

if( !seen[ ady[ x ][ i ] ] ){

dfs( ady[ x ][ i ] , s + (char)(ady[ x ][ i ]+'A') );

}

}

seen[ x ] = 0;

}

int main(){

char line[ 105 ];

string s;

while( gets( line ), line[ 0 ] != '#' ){

s = "";

memset( visited , 0 , sizeof( visited ) );

memset( connected , 0 , sizeof( connected) ) ;

for( int i = 0 ; line[ i ] ; ++i ){

if( line[ i ] == ';' ){

for( int j = 2 ; j < s.length() ; ++j ){

connected[ s[ 0 ] - 'A' ][ s[ j ] - 'A' ] = 1;

connected[ s[ j ] - 'A' ][ s[ 0 ] - 'A' ] = 1;

visited[ s[ j ] - 'A' ] = 1;

}

visited[ s[ 0 ] - 'A' ] = 1;

s = "";

}

else s+= line[ i ];

}

for( int j = 2 ; j < s.length() ; ++j ){

connected[ s[ 0 ] - 'A' ][ s[ j ] - 'A' ] = 1;

connected[ s[ j ] - 'A' ][ s[ 0 ] - 'A' ] = 1;

visited[ s[ j ] - 'A' ] = 1;

}

visited[ s[ 0 ] - 'A' ] = 1;

for( int i = 0 ; i < MAX ; ++i ){

if( !visited[ i ] ) continue;

for( int j = i + 1 ; j < MAX ; ++j ){

if( !visited[ j ] ) continue;

ady[ i ].push\_back( j );

ady[ j ].push\_back( i );

}

}

len = 0;

string ini;

int ini\_;

for( int i = 0 ; i < MAX ; ++i ){

if( visited[ i ] ){

len++;

}

}

mini = 1000;

for( int i = 0 ; i < MAX ; ++i ){

ini = "";ini += (char)( i + 'A' );

if( visited[ i ]){

memset( seen , 0 , sizeof( seen ) );

dfs( i , ini );

}

}

for( int i = 0 ; i < len ; ++i )

printf("%c " , resp[ i ] );

printf("-> %d\n" , mini );

for( int i = 0 ; i < MAX ; ++i )ady[ i ].clear();

}

return 0;

}

# The Dole Queue

\*\*\*ID: 133

\*\*\*Tipo: Simulation

#include <bits/stdc++.h>

bool circle[ 24 ];

int main(){

int k , n , m , next\_left , next\_right , l , r, q , len;

while( scanf("%d %d %d" , &n , &k , &m ) , k| m | n ){

memset( circle , 0 , sizeof( circle ) );

r = 0; l = n - 1;

k--;

m--;

q = 0;

len = n;

while( len ){

for( int j = k % len ; ; r++ ){

if( r == n ) r = 0;

if( !circle[ r ] && j-- <= 0 )break;

}

for( int j = m % len ; ; l-- ){

if( l == -1 ) l = n - 1;

if( !circle[ l ] && j-- <= 0 )break;

}

circle[ r ] = circle[ l ] = 1;

if( q++ )printf(",");

if( r == l ){

printf("%3d" , r + 1 );

len--;

}

else{

printf("%3d%3d" , r + 1 , l + 1 );

len -= 2;

}

}

printf("\n");

}

return 0;

}

# Roman Roulette

\*\*\*ID: 130

\*\*\*Tipo: Simulation, Josephus

#include <bits/stdc++.h>

#include <iostream>

#include <vector>

using namespace std;

#define MAX 105

int main( ){

int n , k , next , kill , k\_next , nn;

while( scanf("%d %d" , &n , &k ) , n|k ){

if( n == 1 ){

printf("1\n");

continue;

}

nn = n;

vector<int> Circle;

for( int i = 0 ; i < n ; ++i )

Circle.push\_back( i );

k--;

for( int i = 0 ; ; ){

kill = ( i + k ) % n;

Circle.erase( Circle.begin() + kill );

n = Circle.size();

k\_next = ( kill + k ) % n;

Circle.insert( Circle.begin() + kill , Circle[ k\_next ] );

if( k\_next >= kill ) k\_next++;

Circle.erase( Circle.begin() + k\_next );

n = Circle.size();

if( n == 1 )break;

if( k\_next >= kill ){

i = ( kill + 1 ) % n;

}

else{

i = kill % n;

}

}

printf("%d\n" , ( ( nn - Circle[ 0 ] ) % nn ) + 1 );

}

return 0;

}

# Numbering Paths

\*\*\*ID: 125

\*\*\*Tipo: Graph Theory, Strongly Connected Components, Transitive Closure

#include <bits/stdc++.h>

#include <string.h>

#define MAX 70

int ady[ MAX ][ MAX ][ MAX ] , V , resp[ MAX ][ MAX ][ MAX ];

void CountingPath(){

int s , i , j , k;

for( s = 2 ; s <= 64 ; ++s ){

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

for( k = 0 ; k < V ; ++k )

ady[ i ][ j ][ s ] += ady[ i ][ k ][ s - 1 ] \* ady[ k ][ j ][ 1 ];

resp[ i ][ j ][ s ] = resp[ i ][ j ][ s - 1 ];

//Si existen rutas mayores al numero de vertices entonces hay ciclo

if( s > V && ady[ i ][ j ][ s ] != 0 ){

resp[ i ][ j ][ s ] = -1;

}else{

resp[ i ][ j ][ s ] += ady[ i ][ j ][ s ];

}

}

}

}

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

( j == 0 ) ? printf("%d" , resp[ i ][ j ][ 64 ] ) : printf(" %d", resp[ i ][ j ][ 64 ]);

}

printf("\n");

}

}

int max( int a , int b ){ return ( a < b )? b:a ;}

int main(){

int n, q = 0 , u , v , i , j;

while( scanf("%d" , &n ) != EOF ){

V = 0;

memset( ady , 0 , sizeof( ady ) );

memset( resp , 0 , sizeof( resp ) );

for( i = 0 ; i < n && scanf("%d %d" , &u , &v ) == 2 ; ++i ){

ady[ u ][ v ][ 1 ]++;

resp[ u ][ v ][ 1 ]++;

V = max( u , max( v , V ) );

}

V++;

printf("matrix for city %d\n" , q++ );

CountingPath();

}

return 0;

}

# The Postal Worker Rings Once

\*\*\*ID: 117

\*\*\*Tipo: Chinese Postman Problem - Euler Tour

#include <bits/stdc++.h>

using namespace std;

#define MAX 105

#define MAXV 27

#define INF 1<<30

struct Edge{

int v , w;

int id;

Edge( int vv , int ww ): v( vv ) , w(ww){}

Edge( int vv , int ww , int i ): v( vv ) , w(ww) , id( i ){}

Edge(){}

};

char s[ MAX ];

int degree[ MAXV ];

vector< Edge > ady[ MAXV ];

int dist[ MAX ] , prev[ MAX ], seen[ MAX ];

void spfa( int source ){

int u , v , w , i;

for( i = 0 ; i < MAXV ; ++i ){

seen[ i ] = 0;

dist[ i ] = INF;

}

queue< int > Q;

dist[ source ] = 0;

Q.push( source );

while( !Q.empty() ){

u = Q.front(); Q.pop();

seen[ u ] = 0;

for( i = 0 ; i < ady[ u ].size() ; ++i ){

v = ady[ u ][ i ].v;

w = ady[ u ][ i ].w;

if( dist[ v ] > dist[ u ] + w ){

dist[ v ] = dist[ u ] + w;

prev[ v ] = u;

if( !seen[ v ] ){

seen[ v ] = 1;

Q.push( v );

}

}

}

}

}

int main(){

long long ans;

int len , source , sink , i;

while( scanf("%s" , &s ) == 1 ){

ans = 0;

len = strlen( s );

if( s[ 0 ] == s[ len - 1 ] ) continue;

memset( degree , 0 , sizeof( degree ) ) ;

ady[ s[ 0 ] - 'a' ].push\_back( Edge( s[ len - 1 ] - 'a' , len ) );

ady[ s[ len - 1 ] - 'a' ].push\_back( Edge( s[ 0 ] - 'a' , len ) );

degree[ s[ 0 ] - 'a' ]++;

degree[ s[ len - 1 ] - 'a' ]++;

ans += len;

while( scanf("%s" , &s ) == 1 ){

len = strlen( s );

if( s[ 0 ] == s[ len - 1 ] ) break;

ady[ s[ 0 ] - 'a' ].push\_back( Edge( s[ len - 1 ] - 'a' , len ) );

ady[ s[ len - 1 ] - 'a' ].push\_back( Edge( s[ 0 ] - 'a' , len ) );

degree[ s[ 0 ] - 'a' ]++;

degree[ s[ len - 1 ] - 'a' ]++;

ans += len;

}

source = sink = -1;

for( i = 0 ; i < MAXV ; ++i ){

if( degree[ i ] & 1 ){

if( source == -1 ) source = i;

else sink = i;

}

}

spfa( source );

ans += dist[ sink ];

printf("%lld\n" , ans );

for( i = 0 ; i < MAXV ; ++i ) ady[ i ].clear();

}

return 0;

}

# Trees on the level

\*\*\*ID: 122

\*\*\*Tipo: Ad hoc, Trees traversal

#include <bits/stdc++.h>

using namespace std;

#define MAX 80005

int tree[ MAX ];

int values[ MAX ];

bool visited[ MAX ];

bool tree\_real[ MAX ];

int Int( string s ){

stringstream ss(s);

int n;

ss>>n;

return n;

}

int main(){

string s;

int nodos = 0;

memset( tree, 0 , sizeof( tree ) );

tree\_real[ 0 ] = true;

while( getline( cin , s) ){

s+=" ";

stringstream ss( s );

bool b = false;

while( ss>>s ){

if(s == "()"){b = true;break;}

int it = s.find( "," );

int nodo = Int( s.substr( 1 , it ) );

string ruta = s.substr( it + 1 );

ruta = ruta.substr(0, ruta.length() - 1 );

int x = 0;

for( int i = 0 ; i < ruta.length() ; ++i ){

if( ruta[ i ] == 'L' )

x = 2 \* x + 1;

else

x = 2 \* x + 2;

tree\_real[ x ] = true;

}

tree[ x ]++;

values[ x ] = nodo;

visited[ x ] = true;

nodos = std::max( nodos , x );

}

if( b ){

int i = 0;

for( ; i <= nodos ; ++i ){

if( tree\_real[ i ] && !visited[ i ] || visited[ i ] && tree[ i ] != 1 ){

cout<<"not complete"<<endl;

break;

}

}

if( i == nodos + 1){

cout<<values[ 0 ];

for( i = 1 ; i <= nodos ; ++i ){

if( values[ i ] )

cout<<" "<<values[ i ];

}

cout<<endl;

}

nodos = 0;

memset( tree , 0 , sizeof( tree ) );

memset( values, 0 , sizeof( values ) );

memset( visited, 0 , sizeof( visited ) );

memset( tree\_real , 0 , sizeof( tree\_real ) );

tree\_real[ 0 ] = true;

}

}

return 0;

}

# Mutant Flatworld Explorers

\*\*\*ID: 118

\*\*\*Tipo: Simulation

#include <bits/stdc++.h>

#define MAX 55

int h , w;

int ady[ MAX ][ MAX ];

char s[ 105 ];

#define N 0

#define W 1

#define S 2

#define E 3

char d[4] = { 'N' , 'W' , 'S' , 'E' };

int dx[4] = { 0 , -1 , 0 , 1 };

int dy[4] = { 1 , 0 , -1 , 0 };

void simulate( int x , int y , char dir ){

for( int i = 0 ; s[ i ] ; ++i ){

if( s[ i ] == 'F' ){

int nx = dx[ dir ] + x, ny = dy[ dir ] + y;

if( nx >= 0 && nx <= h && ny >= 0 && ny <= w ){

x = nx;

y = ny;

}

else{

if( ady[ x ][ y ] ){

continue;

}

else{

printf("%d %d %c LOST\n" , x , y , d[ dir ] );

ady[ x ][ y ] = 1;

return;

}

}

}

else if( s[ i ] == 'L' ){

dir = (dir + 1) % 4;

}

else {

dir = ( 4 + dir - 1 ) % 4;

}

}

printf("%d %d %c\n" , x , y , d[ dir ] );

}

int main(){

int x , y;

char c;

scanf("%d %d" , &h , &w );

memset( ady , 0 , sizeof( ady ) );

while( scanf("%d %d %c" , &x , &y , &c ) != EOF ){

gets( s );

gets( s );

simulate( x , y , ( c=='N')?0:( c == 'S')?2:( c == 'E')?3:1 );

}

return 0;

}

# Stacks of Flapjacks

\*\*\*ID: 120

\*\*\*Tipo: Stacks, Greedy

#include <bits/stdc++.h>

using namespace std;

#define MAX 205

#define MAXN 105

char line[ MAX ];

int a[ MAXN ], b[ MAXN ] , l;

bool cmp( int a , int b ){ return a > b;}

int getIdx( int x ){ return l - x; }

void print(){

for( int i = 0 ; i < l ; ++i ) printf("%d " , a[ i ] ) ;

printf("\n");

}

void flip( int idx ){

int left = 0 , right = idx + 1;

while( left < right ){

int tmp = a[ left ];

a[ left ] = a[ right - 1 ];

a[ right - 1 ] = tmp;

left++; right--;

}

//print();

}

void solve(){

int i , len = l , j;

sort( b , b + l , cmp );

for( i = 0 ; i < l ; ++i ){

for( j = 0 ; j < len ; ++j ){

if( b[ i ] == a[ j ] ) break;

}

if( j != len - 1 ){

if( j == 0 ){

flip( len - 1 );

printf("%d " , l - len + 1 );

}

else {

flip( j );

printf("%d " , l - j );

flip( len - 1 );

printf("%d " , l - len + 1 );

}

}

len--;

}

printf("0\n");

}

int main(){

int x;

while( gets( line ) ){

l = 0;

stringstream ss( line );

while( ss>>x ){ a[ l ] = b[ l++ ] = x; }

for( int i = 0 ; i < l ; ++i ){

if( i ) printf(" ");

printf("%d" , a[ i ] );

}

printf("\n");

solve();

}

return 0;

}

# Arbitrage

\*\*\*ID: 104

\*\*\*Tipo: Floyd Warshall

#include <bits/stdc++.h>

#include <string.h>

#define MAX 25

#define INF 1<<30

int V , path[ MAX ][ MAX ][ MAX ];

double ady[ MAX ][ MAX ][ MAX ];

void init2(){

int i , j;

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

path[ i ][ j ][ 1 ] = i;

}

}

}

//FLOYD 3 dimensiones

//i: vertice origen

//j: vertice destino

//s: numero de aristas entre origen y destino

//ady[ i ][ j ][ s ] = maximo o minimo costo entre i y j usando s pasos.

void floydSteps(){

int i , j , k , s;

for( s = 2 ; s <= V ; ++s ){

for( k = 0 ; k < V ; ++k ){

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

if( ady[ i ][ k ][ s - 1 ] \* ady[ k ][ j ][ 1 ] > ady[ i ][ j ][ s ] ){

ady[ i ][ j ][ s ] = ady[ i ][ k ][ s - 1 ] \* ady[ k ][ j ][ 1 ];

path[ i ][ j ][ s ] = k;

}

}

}

}

}

}

void print( int x , int y , int s ){

if( s == 0 ) printf("%d" , x + 1 );

else{

print( x , path[ x ][ y ][ s ] , s - 1 );

printf(" %d" , y + 1 );

}

return;

}

int main(){

int i , j , s;

double w;

while( scanf("%d" , &V ) != EOF ){

memset( ady , 0 , sizeof( ady ) );

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

if( i == j ) { ady[ i ][ j ][ 1 ] = 1.0;}

else{

scanf("%lf" , &w );

ady[ i ][ j ][ 1 ] = w;

}

}

}

init2();

floydSteps();

for( s = 0 ; s <= V ; ++s ){

for( i = 0 ; i < V ; ++i ){

if( ady[ i ][ i ][ s ] > 1.01 ){

print( i , i , s );

break;

}

}

if( i != V ) break;

}

if( s == V + 1 ) printf("no arbitrage sequence exists");

printf("\n");

}

return 0;

}

# Unidirectional TSP

\*\*\*ID: 116

\*\*\*Tipo: DP

#include <bits/stdc++.h>

using namespace std;

int a[ 21 ][ 105 ], dp[ 21 ][ 105 ];

struct Estado{

int x, y;

Estado( int xx , int yy ):x( xx ) , y( yy ){}

Estado(){}

}prev[ 21 ][ 105 ];

int main(){

int h , w, idx\_d , idx\_u;

while( scanf("%d %d" , &h, &w ) == 2 ){

for( int i = 1 ; i <= h ; ++i ){

for( int j = 1 ; j <= w ; ++j ){

scanf( "%d" , &a[ i ][ j ] );

dp[ i ][ j ] = 1<<30;

}

}

for( int k = 0 ; k <= h ; ++k ) {

dp[ k ][ w ] = a[ k ][ w ];

prev[ k ][ 0 ] = Estado( -1 , - 1 );

}

for( int j = w - 1 ; j >= 1; --j ){

for( int i = 1 ; i <= h ; ++i ){

idx\_d = i + 1;

if( i == h ){

idx\_d = 1;

}

idx\_u = i - 1;

if( i == 1 ){

idx\_u = h;

}

dp[ i ][ j ] = dp[ idx\_d ][ j + 1 ] + a[ i ][ j ];

prev[ i ][ j ] = Estado( idx\_d , j + 1 );

if( dp[ i ][ j ] > dp[ i ][ j + 1 ] + a[ i ][ j ] ){

dp[ i ][ j ] = dp[ i ][ j + 1 ] + a[ i ][ j ];

prev[ i ][ j ] = Estado( i , j + 1 );

}

else if( dp[ i ][ j ] == dp[ i ][ j + 1 ] + a[ i ][ j ] ){

if( prev[ i ][ j ].x > i )

prev[ i ][ j ] = Estado( i , j + 1 );

}

if( dp[ i ][ j ] > dp[ idx\_u ][ j + 1 ] + a[ i ][ j ] ){

dp[ i ][ j ] = dp[ idx\_u ][ j + 1 ] + a[ i ][ j ];

prev[ i ][ j ] = Estado( idx\_u , j + 1 );

}

else if( dp[ i ][ j ] == dp[ idx\_u ][ j + 1 ] + a[ i ][ j ] ){

if( prev[ i ][ j ].x > idx\_u )

prev[ i ][ j ] = Estado( idx\_u , j + 1 );

}

}

}

int min = 1<<30, x = 0 ;

for( int i = 1 ; i <= h ; ++i ){

if( min > dp[ i ][ 1 ] ){

x = i;

min = dp[ i ][ 1 ];

}

}

int resp[ 105 ], len = 0;

for( int i = 0 ; i < w ; ++i ){

resp[ i ] = x;

x = prev[ x ][ i + 1 ].x;

}

printf("%d" , resp[ 0 ] );

for( int i = 1 ; i < w ; ++i ) printf(" %d" , resp[ i ] );

printf("\n%d\n" , min );

}

return 0;

}

# Power Crisis

\*\*\*ID: 151

\*\*\*Tipo: Simulation, Josephus

#include <bits/stdc++.h>

//1 omitimos, empezamos desde 2 como si fuera 0

int survivor(int n, int m){

int i, s;

for (s = 0, i = 1; i <= n; i++){

s = (s + m) % i;

}

return (s + 2);

}

int main(){

int n;

while( scanf("%d" , &n ) && n ){

for( int i = 1 ; ; ++i ){

if( survivor( n - 1 , i ) == 13 ){

printf("%d\n" , i );

break;

}

}

}

return 0;

}

# Traffic Lights

\*\*\*ID: 161

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

#define VERDE 0

#define NARANJA 1

#define ROJO 2

int main(){

int a , b , c, value[ 105 ] , len , light[ 105 ] , green[ 105 ];

while( scanf("%d %d %d" , &a , &b , &c ) , a | b | c ){

value[ 0 ] = a;

value[ 1 ] = b;

light[ 0 ] = light[ 1 ] = VERDE;

green[ 0 ] = a - 5;

green[ 1 ] = b - 5;

len = 2;

while ( c != 0 ){

value[ len ] = c;

green[ len ] = c - 5;

light[ len++ ] = VERDE;

scanf("%d" , &c );

}

int min , i , t;

for( t = 0 ; t <= 18000 ; ){

for( min = green[ 0 ] , i = 1 ; i < len ; ++i ){

if( min > green[ i ] ) min = green[ i ];

}

for( t += min , i = 0 ; i < len ; ++i ){

if( ( green[ i ] -= min ) > 0 )continue;

if( light[ i ] == VERDE ){

light[ i ] = NARANJA;

green[ i ] = 5;

}

else if( light[ i ] == NARANJA ){

light[ i ] = ROJO;

green[ i ] = value[ i ];

}

else{

light[ i ] = VERDE;

green[ i ] = value[ i ] - 5;

}

}

for( i = 0 ; i < len ; ++i ){

if( light[ i ] != VERDE )break;

}

if( i == len )break;

}

if( t > 18000 ){

puts("Signals fail to synchronise in 5 hours");

}

else{

printf("%.2d:%.2d:%.2d\n" , t/3600 , ( t % 3600 )/60 , t % 60 );

}

}

return 0;

}

# Beggar My Neighbour

\*\*\*ID: 162

\*\*\*Tipo: Ad hoc

#include <iostream>

#include <deque>

#include <cstdio>

using namespace std;

int main() {

string tmp;

while(cin >> tmp, tmp[ 0 ] != '#') {

deque<string> A[ 2 ];

A[ 0 ].push\_front(tmp);

for( int i = 1 ; i < 52 ; ++i ) {

cin >> tmp;

A[ i % 2 ].push\_front( tmp );

}

deque<string> T;

int turn = 0, needed = 1, last = 0;

bool flag = 1;

while( flag ) {

while( needed-- ) {

if( !A[ turn ].size()) {

printf("%d%3d\n", ( turn ? 2 : 1 ), A[ ( turn + 1 ) % 2 ].size());

flag= 0;

break;

}

T.push\_back( A[ turn ].front() );

A[ turn ].pop\_front();

int tn = needed;

switch( T.back()[ 1 ] ) {

case 'A':

++needed;

case 'K':

++needed;

case 'Q':

++needed;

case 'J':

++needed;

needed -= tn;

last = turn + 1;

turn = ( turn + 1 )%2;

}

}

if(flag) {

if(last--) {

A[ last ].insert( A[ last ].end(), T.begin(), T.end());

T.clear();

turn = ( last + 1 ) % 2;

}

last = 0;

needed = 1;

turn = ( turn + 1 ) % 2;

}

}

}

return 0;

}

# String Computer

\*\*\*ID: 164

\*\*\*Tipo: Edit Distance - Impresion

#include <bits/stdc++.h>

using namespace std;

#define MAX 50

#define INSERT 0

#define DELETE 1

#define CHANGE 3

#define MATCH 4

char A[ MAX ], B[ MAX ];

int lenA , lenB , dp[ MAX ][ MAX ] , prev[ MAX ][ MAX ], k;

void editDistance(){

lenA = strlen( A );

lenB = strlen( B );

int i , j;

for( i = 0 ; i <= lenA ; ++i ){

dp[ i ][ 0 ] = i;

prev[ i ][ 0 ] = DELETE;

}

for( i = 0 ; i <= lenB ; ++i ){

dp[ 0 ][ i ] = i;

prev[ 0 ][ i ] = INSERT;

}

for( i = 1 ; i <= lenA ; ++i ){

for( j = 1 ; j <= lenB ; ++j ){

if( A[ i - 1 ] == B[ j - 1 ] ){

dp[ i ][ j ] = dp[ i - 1 ][ j - 1 ];

prev[ i ][ j ] = MATCH;

}

else if( dp[ i - 1 ][ j ] > dp[ i ][ j - 1 ] && dp[ i - 1 ][ j - 1 ] > dp[ i ][ j - 1 ] ){

dp[ i ][ j ] = dp[ i ][ j - 1 ] + 1;

prev[ i ][ j ] = INSERT;

}

else if( dp[ i - 1 ][ j ] > dp[ i - 1 ][ j - 1 ]){

dp[ i ][ j ] = dp[ i - 1 ][ j - 1 ] + 1;

prev[ i ][ j ] = CHANGE;

}

else{

dp[ i ][ j ] = dp[ i - 1 ][ j ] + 1;

prev[ i ][ j ] = DELETE;

}

}

}

}

void printEditDistance( int i , int j ){

if( !i && !j ) return;

if( prev[ i ][ j ] == MATCH ){

printEditDistance( i - 1 , j - 1 );

return;

}

if( prev[ i ][ j ] == INSERT ){

printEditDistance( i , j - 1 );

printf("I%c%02d" , B[ j - 1 ] , j );

k--;

}

if( prev[ i ][ j ] == DELETE ){

printEditDistance( i - 1 , j );

printf("D%c%02d" , A[ i - 1 ] , i - k );

k++;

}

if( prev[ i ][ j ] == CHANGE ){

printEditDistance( i - 1 , j - 1 );

printf("C%c%02d" , B[ j - 1 ] , j );

}

}

int main(){

while( scanf("%s" , &A ) , A[ 0 ] != '#' ){

scanf("%s" , &B );

editDistance();

k = 0;

printEditDistance( lenA , lenB );

printf("E\n");

}

return 0;

}

# The Sultan's Successors

\*\*\*ID: 167

\*\*\*Tipo: Backtracking

#include <bits/stdc++.h>

#define MAX 9

int col[ MAX ] , leftDiag[ 20 ] , rightDiag[ 20 ], tmp[ MAX ], a[ MAX ][ MAX ];

int resp[ 100 ][ MAX ] , len = 0;

void gen( int row ){

if( row == 8 ){

for( int i = 0 ; i < 8 ; ++i ) resp[ len ][ i ] = tmp[ i ];

len++;

return;

}

for( int i = 0 ; i < 8 ; ++i ){

if( col[ i ] && leftDiag[ i + row ] && rightDiag[ i - row + 8 ] ){

col[ i ] = leftDiag[ i + row ] = rightDiag[ i - row + 8 ] = 0;

tmp[ row ] = i;

gen( row + 1 );

col[ i ] = leftDiag[ i + row ] = rightDiag[ i - row + 8 ] = 1;

}

}

}

int main(){

len = 0;

memset( col , 1 , sizeof( col ) );

memset( leftDiag , 1 , sizeof( leftDiag ) );

memset( rightDiag , 1 , sizeof( rightDiag ) );

gen( 0 );

int t, maxi, sum;

scanf("%d" , &t );

while( t-- ){

for( int i = 0 ; i < 8 ; ++i )

for( int j = 0 ; j < 8 ; ++j ) scanf("%d" , &a[ i ][ j ] );

maxi = 0;

for( int i = 0 ; i < len ; ++i ){

sum = 0;

for( int j = 0 ; j < 8 ; ++j ){

sum += a[ j ][ resp[ i ][ j ] ];

}

if( sum > maxi ) maxi = sum;

}

printf("%5d\n" , maxi );

}

return 0;

}

# Theseus and the Minotaur

\*\*\*ID: 168

\*\*\*Tipo: DFS

#include <bits/stdc++.h>

using namespace std;

char s[ 260 ];

#define MAX 30

char t , m;

int k;

vector< int > ady[ MAX ];

bool light[ MAX ];

void dfs( int source , int len , int minotaur ){

if( len != 0 && len % k == 0 ){

if( !light[ source ] ) printf("%c " , source + 'A' );

light[ source ] = 1;

}

bool enter = false;

int j , v;

for( j = 0 ; j < ady[ minotaur ].size() ; ++j ){

v = ady[ minotaur ][ j ];

if( !light[ v ] && v != source && v != minotaur ){

enter = true;

dfs( minotaur , len + 1 , v );

break;

}

}

if( !enter ){

printf("/%c\n" , minotaur + 'A' );

return;

}

}

int main(){

int len , i , j , cnt;

while( scanf("%s" , &s ) , s[ 0 ] != '#'){

len = strlen( s );

for( i = 0 ; i < len ; ++i ) if( s[ i ] == ';' || s[ i ] == '.' ) s[ i ] = ' ';

scanf(" %c %c %d" , &m , &t ,&k );

stringstream ss( s );

while( ss>>s ){

for( i = 2 ; s[ i ] ; ++i ){

ady[ s[ 0 ] - 'A' ].push\_back( s[ i ] - 'A' );

}

}

memset( light , 0 , sizeof( light ) );

dfs( t - 'A' , 0 , m - 'A' );

for( i = 0 ; i < MAX ; ++i ) ady[ i ].clear();

}

return 0;

}

# Clock Patience

\*\*\*ID: 170

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

char rank[ 14 ] = { ' ','A', '2', '3', '4', '5', '6', '7', '8', '9', 'T', 'J', 'Q', 'K' };

char suit[ 4 ] = { 'S' , 'H' , 'D' , 'C' };

queue<string> deck[ 14 ];

int main(){

int pos, ans;

char s[ 3 ];

string act;

while( scanf("%s" , &s ) , s[ 0 ] != '#' ){

deck[ 13 ].push( s );

pos = 12;

for( int i = 0 ; i < 51 ; ++i ){

scanf("%s" , s );

deck[ pos-- ].push( s );

if( pos == 0 ) pos = 13;

}

int current = 13;

ans = 0;

while( 1 ){

if( deck[ current ].empty() )break;

act = deck[ current ].front();

deck[ current ].pop();

for( int i = 1 ; i < 14 ; ++i ){

if( act[ 0 ] == rank[ i ] ){

current = i;

break;

}

}

ans++;

}

printf("%.2d,%s\n" , ans , act.c\_str() );

for( int i = 0 ; i < 14 ; ++i ){

while( !deck[ i ].empty() ){

deck[ i ].pop();

}

}

}

return 0;

}

# Eeny Meeny

\*\*\*ID: 180

\*\*\*Tipo: Simulation, Josephus

#include <bits/stdc++.h>

using namespace std;

int survivor( int n , int m ){

int s = 0;

for( int i = 1 ; i <= n ; ++i ) s= ( s + m ) % i;

return s + 1;

}

int main(){

int l , u, mini, s, resp;

while( scanf("%d %d" , &l , &u ) && l|u ){

mini = 1<<30;

for( int i = l ; i <= u ; ++i ){

s = survivor( i , 15 );

cout<<s<<endl;

if( abs( 1 - s ) < mini ){ mini = abs( 1 - s ); resp = s; }

}

cout<<resp<<endl;

}

return 0;

}

# Laser Lines

\*\*\*ID: 184

\*\*\*Tipo: Geometria

#include <bits/stdc++.h>

#include<stdio.h>

#include<map>

#include<vector>

#include<math.h>

#include<cstring>

#include<algorithm>

using namespace std;

#define MAX 303

struct Point{

int x , y;

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

Point(){}

}v[ MAX ];

vector<Point> line[ MAX ];

bool cmp( Point p1 , Point p2 ){

return ( p1.x < p2.x || ( p1.x == p2.x && p1.y < p2.y ) );

}

bool f( Point p1 , Point p2 ){

return ( p1.x == p2.x && p1.y == p2.y );

}

int main(){

int len , cnt , tam ;

bool seen[ MAX ][ MAX ];

while( scanf("%d %d" , &v[ 0 ].x , &v[ 0 ].y ) , v[ 0 ].x | v[ 0 ].y ){

len = 1;

memset( seen , 0 , sizeof( seen ) );

while( scanf("%d %d" , &v[ len ].x , &v[ len ].y ), v[ len ].x | v[ len++ ].y );

len--;

sort( v , v + len , cmp );

len = unique( v , v+len , f ) - v;

tam = 0;

for( int i = 0 ; i < len ; ++i ){

for( int j = i + 1 ; j < len ; ++j ){

cnt = 2;

if( seen[ i ][ j ] )continue;

int k = 0;

for( ; k < j ; ++k ){

if( k != i &&

(( v[k].y - v[i].y ) \* ( v[j].x - v[i].x ) - ( v[k].x - v[i].x )\*( v[j].y - v[i].y ) == 0)){

k = -1;

break;

}

}

if( k == -1 )continue;

for( k = j + 1 ; k < len ; ++k ){

if( ( v[ k ].y - v[ i ].y ) \* ( v[ j ].x - v[ i ].x ) - ( v[ k ].x - v[ i ].x )\*( v[ j ].y - v[ i ].y ) == 0){

cnt++;

seen[ i ][ k ] = seen[ j ][ k ] = 1;

line[ tam ].push\_back( Point( v[ k ].x , v[ k ].y ) );

}

}

if( cnt >= 3 ){

line[ tam ].push\_back( Point( v[ i ].x , v[ i ].y ) );

line[ tam ].push\_back( Point( v[ j ].x , v[ j ].y ) );

tam++;

}

seen[ i ][ j ] = 1;

}

}

if( tam == 0 ) puts("No lines were found");

else{

puts("The following lines were found:");

for( int i = 0 ; i < tam ; ++i ){

printf("(%4d,%4d)(%4d,%4d)",

line[ i ][ line[ i ].size() - 2 ].x,

line[ i ][ line[ i ].size() - 2 ].y,

line[ i ][ line[ i ].size() - 1 ].x,

line[ i ][ line[ i ].size() - 1 ].y);

for( int j = 0 ; j < line[ i ].size() - 2 ; ++j ){

printf("(%4d,%4d)" , line[ i ][ j ].x,line[ i ][ j ].y);

}

putchar('\n');

}

}

for( int i = 0 ; i < tam ; ++i )line[ i ].clear();

}

}

# Trip Routing

\*\*\*ID: 186

\*\*\*Tipo: Graph Theory, Floyd Warshall

#include <bits/stdc++.h>

using namespace std;

#define MAX 300

#define INF 9999999

int ady[ MAX ][ MAX ] , costos[ MAX ][ MAX];

int V;

int path[ MAX ][ MAX ];

void Init(){

for(int i = 0 ; i < MAX ; ++i){

for(int j = 0 ; j < MAX; ++j){

ady[ i ][ j ] = INF;

}

ady[ i ][ i ] = 0;

}

}

void InitPath(){

for(int i = 0 ; i < MAX ; ++i ){

for(int j = 0 ; j < MAX ; ++j ){

if( ady[ i ][ j ] == INF || ady[ i ][ j ] == 0 ){ path[ i ][ j ] = 0; }

else path[ i ][ j ] = i;

}

}

}

void floyd(){

for( int k = 0 ; k < V ; ++k ){

for( int i = 0 ; i < V ; ++i ){

for( int j = 0 ; j < V ; ++j ){

int t = ady[ i ][ k ] + ady[ k ][ j ];

if( t < ady[ i ][ j ] ){

ady[ i ][ j ] = t;

path[ i ][ j ] = path[ k ][ j ];

}

}

}

}

}

vector<int> print;

void ruta( int x , int y ){

if( x == INF || y == INF )return;

if( x == y )print.push\_back( x );

else{

ruta( x , path[ x ][ y ] );

print.push\_back( y );

}

return;

}

int toInt( string s ){ stringstream ss( s ); int n ; ss>>n; return n;}

int main(){

char line[ MAX ] , u[ 30 ] , v [ 30 ] , route[ 15 ];

int w;

map<string ,int > mp;

map< int, string > mp2;

string rutas[ MAX ][ MAX ];

V = 0;

string uu ,vv , rr , pp;

Init();

while( gets( line ) ){

int len = strlen( line );

if( len == 0 )break;

uu =""; vv="";rr=""; pp="";

int i = 0;

for( ; i < len ; ++i ){

if( line[ i ] == ',' )break;

uu += line[ i ];

}

i++;

for( ; i < len ; ++i ){

if( line[ i ] == ',' )break;

vv += line[ i ];

}

strcat( route , "" );

i++;

for( ; i < len ; ++i ){

if( line[ i ] == ',' )break;

rr += line[ i ];

}

i++;

for( ; i < len ; ++i )pp += line[ i ];

if( mp.find( uu ) == mp.end() ){

mp2[ V ] = uu;

mp[ uu ] = V++;

}

if( mp.find( vv ) == mp.end() ){

mp2[ V ] = vv;

mp[ vv ] = V++;

}

if( ady[ mp[ uu ] ][ mp[ vv ] ] < toInt( pp ) )continue;

ady[ mp[ uu ] ][ mp[ vv ] ] = ady[ mp[ vv ] ][ mp[ uu ] ] = toInt( pp );

costos[ mp[ uu ] ][ mp[ vv ] ] = costos[ mp[ vv ] ][ mp[ uu ] ] = toInt( pp );

rutas[ mp[ uu ] ][ mp[ vv ] ] = rutas[ mp[ vv ] ][ mp[ uu ] ] =rr ;

}

InitPath();

floyd();

while( gets( line ) ){

int len = strlen( line );

uu = "";

vv = "";

int i = 0;

for( ; i < len ; ++i ){

if( line[ i ] == ',' )break;

uu += line[ i ];

}

i++;

for( ; i < len ; ++i )vv += line[ i ];

printf("\n\n");

printf("From To Route Miles\n");

printf("-------------------- -------------------- ---------- -----\n");

print.clear();

ruta( mp[ uu ] , mp[ vv ] );

for( i = 0 ; i < print.size() - 1; ++i ){

printf("%-20s %-20s %-10s %5d\n" , mp2[ print[ i ] ].c\_str() , mp2[ print[ i + 1 ] ].c\_str() ,

rutas[ print[ i ] ][ print[ i + 1 ] ].c\_str() , costos[ print[ i ] ][ print[ i + 1 ] ]);

}

printf(" -----\n");

printf(" Total %5d\n" , ady[ mp[ uu ] ][ mp[ vv ] ]);

}

return 0;

}

# Anagram

\*\*\*ID: 195

\*\*\*Tipo: Recursion, sorting

#include <bits/stdc++.h>

#include <iostream>

using namespace std;

#define MAX 105

bool seen[ MAX ];

char s[ MAX ];

int l;

char resp[ MAX ];

int cmp(char a , char b)

{

if( islower( a ) && islower( b ) ) return a < b;

if( isupper( a ) && isupper( b ) ) return a < b;

if( islower( a ) && isupper( b ) ) return a < ( b - 'A' + 'a');

if( isupper( a ) && islower( b ) ) return a <= ( b - 'a' + 'A');

}

void solve( int len ){

if( len == l ){

printf("%s\n" , resp );

return;

}

for( int i = 0 ; i < l ; ++i ){

if( seen[ i ] )continue;

resp[ len ] = s[ i ];

seen[ i ] = 1;

solve( len + 1 );

seen[ i ] = 0;

while( i < (l - 1) && s[ i ] == s[ i + 1 ] )i++;

}

}

int main(){

int t ;

scanf( "%d" , &t );

while( t-- ){

scanf("%s" , &s );

l = strlen( s );

sort( s , s + l , cmp );

resp[ l ] = '\0';

memset( seen , 0 , sizeof( seen ) );

solve( 0 );

}

return 0;

}

# Rare Order

\*\*\*ID: 200

\*\*\*Tipo: Graph Theory, Topological Sort

#include <bits/stdc++.h>

#include <sstream>

#include <vector>

#include <queue>

using namespace std;

#define MAX 35

int used[ MAX ];

///reglas representadas con entero a < b -> [ 0 ][ 1 ] = true

int inDegree[MAX];

vector <int> ady[ MAX];

char let[ MAX ];

int idx;

char output[ MAX + 5 ];

void TopologicalSort(){

queue <int> Q;

for ( int i = 0; let[ i ] ; ++i ) {

if ( inDegree [ let[ i ] -'A' ] == 0 )

Q.push ( let[ i ] -'A' );

}

vector <int> sortedList;

while ( !Q.empty () ) {

int pop = Q.front ();

Q.pop ();

sortedList.push\_back (pop);

//para adyacentes de nodo actual

for ( unsigned int i = 0; i < ady[pop].size (); ++i ) {

//reducimos grado de adyacentes luego de haber eliminado nodo q los apuntaba

inDegree [ ady[ pop ] [ i ] ]--;

if ( inDegree [ ady[ pop ] [ i ] ] == 0 )

Q.push ( ady[ pop ][ i ]);

}

}

for ( int i = sortedList.size() - 1 ; i >= 0 ; --i )

printf("%c",sortedList[ i ] + 'A' );

printf("\n");

}

int main(){

char s[ MAX ], line[ MAX ];

memset( used, 0, sizeof( used ) );

memset (inDegree, 0, sizeof( inDegree) );

scanf("%s", s );

if( !strcmp( s, "#") )return 0;

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

used[ s[ i ] - 'A' ] = 1;

while( scanf("%s", line) && strcmp( line, "#") ){

for( int i = 0 ; s[ i ] && line[ i ] ; ++i ){

used[ line[ i ] - 'A' ] = 1;

if( s[ i ] != line[ i ]){

ady[ line[ i ] -'A' ].push\_back( s[ i ] -'A');

inDegree[ s[ i ] - 'A' ]++;

break;

}

}

strcpy( s , line );

}

idx = 0;

for( int i = 0 ; i < MAX ; ++i )

if( used[ i ] ) { let[ idx ] = i + 'A' ; used[ idx++ ] = 0;}

TopologicalSort();

return 0;

}

# FireTruck

\*\*\*ID: 208

\*\*\*Tipo: DFS

#include <bits/stdc++.h>

using namespace std;

int V , cnt , end ;

#define MAX 25

int ady[ MAX ][ MAX ], G[ MAX ][ MAX ];

map<int , bool> \_seen;

bool seen[ MAX ];

int path[ MAX ];

void TransitiveClosure(){

for( int k = 1; k <= V ; ++k )

for( int i = 1 ; i <= V ; ++i )

for( int j = 1 ; j <= V ; ++j )

ady[ i ][ j ] |= ( ady[ i ][ k ] & ady[ k ][ j ] );

}

void solve( int u , int len ){

path[ len ] = u;

if( u == end ){

for( int i = 0 ; i <= len ; ++i ){

if( i )

printf(" ");

printf("%d" , path[i]);

}

printf("\n");

cnt++;

return;

}

for( int i = 1 ; i <= V ; ++i ){

if( !seen[ i ] && ady[ i ][ end ] && G[u][i] ){

seen[ i ] = 1;

solve( i , len + 1 );

seen[ i ] = 0;

}

}

}

int main(){

int t = 1 , u , v;

while( scanf("%d" , &end ) != EOF ){

memset( ady , 0 , sizeof( ady ) );

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

while( scanf("%d %d" , &u , &v ) , u|v ){

ady[ u ][ v ] = ady[ v ][ u ] = 1;

G[u][v] = G[v][u] = 1;

V= max( V , max(u , v) );

}

TransitiveClosure();

printf("CASE %d:\n" , t++ );

memset( seen , 0 , sizeof( seen ) );

cnt = 0;

seen[ 1 ] = 1;

path[0] = 1;

solve( 1 , 0 );

printf("There are %d routes from the firestation to streetcorner %d.\n" , cnt , end );

}

return 0;

}

# Getting in Line

\*\*\*ID: 216

\*\*\*Tipo: TSP, DP

#include <bits/stdc++.h>

using namespace std;

#define MAX 10

int x[ MAX ] , y[ MAX ] , n;

int prev[ MAX ];

double d[ MAX ][ MAX ];

double dist( int i , int j ){

return sqrt( ( x[ i ] - x[ j ] ) \* ( x[ i ] - x[ j ] ) + ( y[ i ] - y[ j ] ) \* ( y[ i ] - y[ j ] ) );

}

int bitcount( int n ){

int cnt = 0;

while( n ){

n = n & ( n - 1 );

cnt++;

}

return cnt;

}

int Last;

double mini;

void print( int last ){

if( prev[ last ] == -1 ){

return;

}

print( prev[ last ] );

printf("Cable requirement to connect (%d,%d) to (%d,%d) is %.2lf feet.\n" , x[ prev[ last ] ] , y [ prev[ last ] ] , x[ last ] , y[ last ] , dist( last , prev[ last ] ) + 16 );

}

int p[ MAX ];

void solve( int last , int mask , double sum ){

if( bitcount( mask ) == n ){

if( sum < mini ){

mini = sum;

Last = last;

memcpy( prev , p , sizeof( p ) );

}

return;

}

for( int j = 0 ; j < n ; ++j ){

if( !( mask & 1<<j ) ){

p[ j ] = last;

solve( j , mask | 1<<j , sum + dist( j , last ) + 16 );

}

}

}

int main(){

int t = 1;

double ans;

while( scanf("%d" , &n ) , n ){

for( int i = 0 ; i < n ; ++i ){

scanf("%d %d" , &x[ i ] , &y[ i ] );

}

puts("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

printf("Network #%d\n" , t++ );

mini = 1<<30;

for( int i = 0 ; i < n ; ++i ){

memset( p , -1 , sizeof( p ) );

solve( i , 1<<i , 0.0 );

}

print( Last );

printf("Number of feet of cable required is %.2lf.\n" , mini );

}

return 0;

}

# Othello

\*\*\*ID: 220

\*\*\*Tipo: Ad hoc - Simulation

#include <bits/stdc++.h>

using namespace std;

int n = 8;

char ady[ 9 ][ 9 ];

int dx[ 8 ] = { 1 , -1 , 0 , 0 , 1 , 1 , -1 , -1 };

int dy[ 8 ] = { 0 , 0 , 1 , -1 , 1 , -1 , 1 , -1};

char player;

bool possible( int x , int y ){

char oponent = ( player == 'B')?'W':'B';

if( ady[ x ][ y ] != '-' )return 0;

int nx , ny;

for( int i = 0 ; i < 8 ; ++i ){

int k = 1, cnt = 0;

while( 1 ){

nx = k \* dx[ i ] + x; ny = k \* dy[ i ] + y;

if( nx >= 0 && ny >= 0 && nx < n && ny < n && ady[ nx ][ ny ] == oponent ){

cnt++;

}

else break;

k++;

}

if( cnt > 0 && ady[ nx ][ ny ] == player ) return 1;

}

return 0;

}

struct Point{

int x , y;

Point(){}

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

}points[ 105 ];

void change( int x , int y ){

char oponent = ( player == 'B')?'W':'B';

if( !possible( x , y ) ) return;

ady[ x ][ y ] = player;

int nx , ny;

for( int i = 0 ; i < 8 ; ++i ){

int k = 1, cnt = 0;

while( 1 ){

nx = k \* dx[ i ] + x; ny = k \* dy[ i ] + y;

if( nx >= 0 && ny >= 0 && nx < n && ny < n && ady[ nx ][ ny ] == oponent ){

points[ cnt++ ] = Point( nx , ny );

}

else break;

k++;

}

if( cnt > 0 && ady[ nx ][ ny ] == player ){

for( int i = 0 ; i < cnt ; ++i ){

ady[ points[ i ].x ][ points[ i ].y ] = player;

}

}

}

int b = 0 , w = 0;

for( int i = 0 ; i < n ; ++i ){

for( int j = 0 ; j < n ; ++j ){

if( ady[ i ][ j ] == 'B')b++;

else if( ady[ i ][ j ] =='W')w++;

}

}

printf("Black - %2d White - %2d\n" , b , w );

}

int main(){

int t;

char Q[ 5 ];

bool first;

scanf("%d" , &t );

for( int q = 0 ; q < t ; ++q){

if( q )printf("\n");

for( int i = 0 ; i < 8 ; ++i )

for( int j = 0 ; j < n ; ++j ) scanf( " %c" , &ady[ i ][ j ] );

scanf(" %c" , &player );

while( scanf("%s" , &Q ), Q[ 0 ] != 'Q' ){

if( Q[ 0 ] == 'L'){

first = true;

for( int i = 0 ; i < n ; ++i ){

for( int j = 0 ; j < n ; ++j ){

if( possible( i , j ) ){

if( !first )printf(" ");

printf("(%d,%d)" , i + 1 , j + 1 );

first = false;

}

}

}

if( !first )printf("\n");

else puts("No legal move.");

}

else if( Q[ 0 ] == 'M'){

int x = ( Q[ 1 ] - '0' ) - 1 , y = ( Q[ 2 ] - '0') - 1;

if( !possible( x , y ) ) player = ( player == 'B') ? 'W' : 'B';

change( x , y );

player = ( player == 'B')?'W':'B';

}

}

for( int i = 0 ; i < n ; ++i ){

for( int j = 0 ; j < n ; ++j ){

printf("%c" , ady[ i ][ j ] );

}

printf("\n");

}

}

return 0;

}

# Correct Move

\*\*\*ID: 255

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

int table[ 64 ][ 64 ];

int dx[ 4 ] = { 1 , -1 , 0 , 0 };

int dy[ 4 ] = { 0 , 0 , 1 , -1 };

int w = 8;

int main(){

int k , q , Q, xk , yk , xq , yq , x , y;

bool stop;

while( scanf("%d %d %d" , &k ,&q , &Q ) != EOF ){

if( k == q ){

puts("Illegal state");

continue;

}

stop = true;

yq = q % w;

xq = ( q - yq ) / w;

yk = k % w;

xk = ( k - yk ) / w;

memset( table , 0 , sizeof( table ) );

for( int i = 0 ; i < 4 ; ++i ){

int nx = xk + dx[ i ], ny = yk + dy[ i ];

if( nx >= 0 && ny >= 0 && nx < w && ny < w ){

table[ nx ][ ny ] = 'K';

}

}

for( int i = 0 ; i < 4 ; ++i ){

int c = 1;

while( 1 ){

int nx = xq + c \* dx[ i ], ny = yq + c \* dy[ i ];

if( nx == xk && ny == yk )break;

if( nx >= 0 && ny >= 0 && nx < w && ny < w ){

if( table[ nx ][ ny ] == 'K'){

table[ nx ][ ny ] = 'A'; //not allowed

}

else table[ nx ][ ny ] = 'Q';

}

else break;

c++;

}

}

y = Q % w;

x = ( Q - y ) / w;

table[ xk ][ yk ] = 0;

table[ xq ][ yq ] = 0;

if( table[ x ][ y ] == 'K' || !table[ x ][ y ] ) puts("Illegal move");

else if( table[ x ][ y ] == 'A' )puts("Move not allowed");

else if( ( k == 0 && Q == 9) || ( k == 7 && Q == 14) ||

( k == 56 && Q == 49 ) || (k == 63 && Q == 54)) printf("Stop\n");

else printf("Continue\n");

}

return 0;

}

# Puzzle

\*\*\*ID: 227

\*\*\*Tipo: Ad hoc , simulation

#include <bits/stdc++.h>

using namespace std;

//down , up , right , left

int dx[ 4 ] = { 1 , -1 , 0 , 0 };

int dy[ 4 ] = { 0 , 0 , 1 , -1 };

int main(){

char ady[ 5 ][ 5 ];

char s[ 505 ] , ss[ 105 ];

bool illegal;

char C;

int x , y , len , nx , ny , q = 0;

while( gets( ady[ 0 ] ) && ady[ 0 ][ 0 ] != 'Z' ){

if( q ){

gets( ady[ 0 ] );

if( ady[ 0 ][ 0 ] == 'Z')break;

printf("\n");

}

for( int i = 1 ; i < 5 ; ++i ) gets( ady[ i ] );

s[ 0 ] = '\0';

while( scanf("%s" , &ss ) ){

strcat( s , ss );

if( ss[ strlen( ss ) - 1 ] == '0')break;

}

len = strlen( s );

for( int i = 0 ; i < 5 ; ++i ){

for( int j = 0 ; j < 5 ; ++ j ){

if( ady[ i ][ j ] == ' '){

x = i; y = j;

i = 5;

break;

}

}

}

illegal = false;

for( int i = 0 ; i < len ; ++i ){

if( s[ i ] == '0' )break;

if( s[ i ] == 'A'){

nx = dx[ 1 ] + x; ny = dy[ 1 ] + y;

if( nx >= 0 && ny >= 0 && nx < 5 && ny < 5 ){

char tmp = ady[ nx ][ ny ];

ady[ nx ][ ny ] = ady[ x ][ y ];

ady[ x ][ y ] = tmp;

x = nx; y = ny;

}

else{

illegal = true;

break;

}

}

else if( s[ i ] == 'B'){

nx = dx[ 0 ] + x; ny = dy[ 0 ] + y;

if( nx >= 0 && ny >= 0 && nx < 5 && ny < 5 ){

char tmp = ady[ nx ][ ny ];

ady[ nx ][ ny ] = ady[ x ][ y ];

ady[ x ][ y ] = tmp;

x = nx; y = ny;

}

else{

illegal = true;

break;

}

}

else if( s[ i ] == 'L'){

nx = dx[ 3 ] + x; ny = dy[ 3 ] + y;

if( nx >= 0 && ny >= 0 && nx < 5 && ny < 5 ){

char tmp = ady[ nx ][ ny ];

ady[ nx ][ ny ] = ady[ x ][ y ];

ady[ x ][ y ] = tmp;

x = nx; y = ny;

}

else{

illegal = true;

break;

}

}

else if( s[ i ] == 'R'){

nx = dx[ 2 ] + x; ny = dy[ 2 ] + y;

if( nx >= 0 && ny >= 0 && nx < 5 && ny < 5 ){

char tmp = ady[ nx ][ ny ];

ady[ nx ][ ny ] = ady[ x ][ y ];

ady[ x ][ y ] = tmp;

x = nx; y = ny;

}

else{

illegal = true;

break;

}

}

}

printf("Puzzle #%d:\n" , ++q );

if( illegal )puts("This puzzle has no final configuration.");

else{

for( int i = 0 ; i < 5 ; ++i ){

for( int j = 0 ; j < 5 ; ++j ){

if( j == 0 )printf("%c" , ady[ i ][ j ] );

else printf(" %c" , ady[ i ][ j ] );

}

printf("\n");

}

}

}

return 0;

}

# Testing the CATCHER

\*\*\*ID: 231

\*\*\*Tipo: DP, LIS

#include<bits/stdc++.h>

using namespace std;

#define N 100002

int B[N], f[N], a[N];

int n,t;

void LIS() {

int i, mid, right, left, num, blen, max = 0;

blen = 1;

B[0] = a[0];

for (i = 0; i < n; i++) {

num = a[i], left = 0, right = blen;

while (left <= right) {

mid = (left + right) / 2;

if (B[mid] < num) left = mid + 1;

else right = mid - 1;

}

f[i] = left;

B[left] = num;

if (blen < left) blen = left;

if (max < f[i]) max = f[i];

}

printf("Test #%d:\n",t++);

printf(" maximum possible interceptions: %d\n", max);

}

int main() {

int x;

t=1;

while (scanf("%d", &x) && x!=-1){

if(t-1)printf("\n");

n=0;

a[n++]=x;

while(scanf("%d",&x) && x!=-1){

a[n++]=x;

}

reverse(a,a+n);

LIS();

}

return 0;

}

# Crossword Answers

\*\*\*ID: 232

\*\*\*Tipo: Ad hoc simulation

#include <bits/stdc++.h>

using namespace std;

#define MAX 105

char ady[ MAX ][ MAX ];

int num[ MAX ][ MAX ];

int h ,w;

int dx[ 2 ] = { -1 , 0 };

int dy[ 2 ] = { 0 , -1 };

void Across(){

for( int i = 0 ; i < h ; ++i ){

for( int j = 0 ; j < w ; ++j ){

if( num[ i ][ j ] != 0 ){

printf("%3d.%c" , num[ i ][ j ] , ady[ i ][ j ] );

while( ady[ i ][ ++j ] != '\*' ){

printf("%c" , ady[ i ][ j ] );

}

printf("\n");

}

}

}

}

void Down(){

for( int i = 0 ; i < h ; ++i ){

for( int j = 0 ; j < w ; ++j ){

if( num[ i ][ j ] != 0 && ( ( i == 0 ) || (i > 0 && ady[ i - 1 ][ j ] == '\*') )){

printf("%3d.%c" , num[ i ][ j ] , ady[ i ][ j ] );

int x = i;

while( ady[ ++x ][ j ] != '\*' ){

printf("%c" , ady[ x ][ j ] );

}

printf("\n");

}

}

}

}

int main(){

int n , q = 0;

while( scanf("%d" , &h ) , h ){

scanf("%d" , &w );

n = 1;

if( q )printf("\n");

for( int i = 0 ; i < h ; ++i ) scanf("%s" , &ady[ i ] );

memset( num , 0 , sizeof( num ) );

for( int i = 0 ; i < h ; ++i ){

for( int j = 0 ; j < w ; ++j ){

if( ady[ i ][ j ] == '\*' )continue;

for( int k = 0 ; k < 2 ; ++k ){

int nx = dx[ k ] + i, ny = dy[ k ] + j;

if( nx >= 0 && ny >= 0 && nx < h && ny < w ){

if( ady[ nx ][ ny ] == '\*'){ num[ i ][ j ] = n++; break;}

}

else{ num[ i ][ j ] = n++; break; }

}

}

}

for( int i = 0 ; i < h ; ++i ) ady[ i ][ w ] = '\*';

for( int i = 0 ; i < w ; ++i ) ady[ h ][ i ] = '\*';

printf("puzzle #%d:\n" , ++q );

printf("Across\n");

Across();

printf("Down\n");

Down();

}

return 0;

}

# Calling Circles

\*\*\*ID: 247

\*\*\*Tipo: Graph Theory, Strongly Connected Components

#include <bits/stdc++.h>

using namespace std;

#define MAX 50

vector<int> ady[ MAX ];

int dfsNum[ MAX ] , lowlink[ MAX ] , scc[ MAX ], num\_scc, V , idx;

bool in\_stack[ MAX ];

stack<int> S;

void tarjan( int v ){

dfsNum[ v ] = lowlink[ v ] = idx++;

S.push( v ); in\_stack[ v ] = true;

for( int i = 0 ; i < ady[ v ].size(); ++i ){

int w = ady[ v ][ i ];

if( dfsNum[ w ] == -1 ){

tarjan( w );

lowlink[ v ] = min( lowlink[ v ] , lowlink[ w ] );

}else if( in\_stack[ w ] )lowlink[ v ] = min( lowlink[ v ] , lowlink[ w ] );

}

if( dfsNum[ v ] == lowlink[ v ] ){

int aux;

do{

aux = S.top(); S.pop();

scc[ aux ] = num\_scc;

in\_stack[ aux ] = false;

}while( aux != v );

num\_scc++;

}

}

void build\_scc(){

memset( dfsNum , -1 , sizeof( dfsNum ) );

memset( in\_stack , 0 , sizeof( in\_stack ) );

idx = num\_scc = 0;

for( int i = 0 ; i < V ; ++i ){

if( dfsNum[ i ] == -1 )tarjan( i );

}

}

int main(){

int E, t = 1;

char s1[ 30 ] , s2[ 30 ];

while( scanf("%d %d",&V,&E) && V|E ){

if( t > 1 )printf("\n");

map<string , int > mp;

map<int , string > mp2;

for( int i = 0 ; i < V ; ++i )ady[ i ].clear();

V = 0;

while( E-- ){

scanf("%s %s", s1 , s2 );

if( mp.find( s1 ) == mp.end() ){

mp2[ V ] = s1;

mp[ s1 ] = V++;

}

if( mp.find( s2 ) == mp.end() ){

mp2[ V ] = s2;

mp[ s2 ] = V++;

}

ady[ mp[ s1 ] ].push\_back( mp[ s2 ] );

}

build\_scc();

printf("Calling circles for data set %d:\n", t++);

for( int i = 0; i < num\_scc ; ++i ){

bool first = true;

for( int j = 0 ; j < V ; ++j ){

if( scc[ j ] == i ){

( first )? printf( "%s", mp2[ j ].c\_str() ) : printf(", %s" , mp2[ j ].c\_str() );

first = false;

}

}

printf("\n");

}

}

return 0;

}

# Cube Painting

\*\*\*ID: 253

\*\*\*Tipo: Ad hoc

//viendo un dado nos damos cuenta d las rotaciones

//el dado q se otorga es:

/\*

1

4 2 3

5

6

se comprubea cada dado y su inverso

6

3 5 4 -> aqui tenemos que invertir los lados xq sino no corresponderia al dado inicial x ello intercambiamos

2

1

6

4 5 3

2

1

\*/

#include <bits/stdc++.h>

#include <string.h>

char cube[ 20 ] , a[ 10 ] , b[ 10 ] , s[ 10 ];

int rot[ 30 ][ 10 ] = { { 1 , 2 , 3 , 4 , 5 , 6 } , { 1 , 3 , 5 , 2 , 4 , 6 } , { 1 , 4 , 2 , 5 , 3 , 6 } , { 1 , 5 , 4 , 3 , 2 , 6 } ,

{ 2 , 1 , 4 , 3 , 6 , 5 } , { 2 , 3 , 1 , 6 , 4 , 5 } , { 2 , 4 , 6 , 1 , 3 , 5 } , { 2 , 6 , 3 , 4 , 1 , 5 } ,

{ 3 , 1 , 2 , 5 , 6 , 4 } , { 3 , 2 , 6 , 1 , 5 , 4 } , { 3 , 5 , 1 , 6 , 2 , 4 } , { 3 , 6 , 5 , 2 , 1 , 4 } };

bool rotate(){

char tmp;

for( int i = 0 ; i < 12 ; ++i ){

for( int j = 0 ; j < 6 ; ++j ) s[ j ] = a[ rot[ i ][ j ] - 1 ];

s[ 6 ] = '\0';

if( !strcmp( s , b ) ) return true;

for( int j = 5 ; j >= 0 ; --j ) s[ 5 - j ] = a[ rot[ i ][ j ] - 1 ];

tmp = s[ 2 ]; s[ 2 ] = s[ 3 ]; s[ 3 ] = tmp;

if( !strcmp( s , b ) ) return true;

}

return false;

}

int main(){

while( scanf("%s" , &s ) == 1 ){

int i = 0;

for( ; i < 6 ; ++i ) a[ i ] = s[ i ];

for( ; i < 12 ; ++i ) b[ i - 6 ] = s[ i ];

a[ 6 ] = '\0'; b[ 6 ] = a[ 6 ];

if( rotate() ) puts("TRUE");

else puts("FALSE");

}

return 0;

}

# Quicksome Squares

\*\*\*ID: 256

\*\*\*Tipo: Brute Force

#include <bits/stdc++.h>

int main(){

int n;

while ( scanf("%d", &n ) > 0 )

if ( n == 2 )

printf("00\n01\n81\n");

else if ( n == 4 )

printf("0000\n0001\n2025\n3025\n9801\n");

else if ( n == 6 )

printf("000000\n000001\n088209\n494209\n998001\n");

else if ( n == 8 )

printf(

"00000000\n00000001\n04941729\n07441984\n"

"24502500\n25502500\n52881984\n60481729\n"

"99980001\n"

);

return 0;

}

# Software Allocation

\*\*\*ID: 259

\*\*\*Tipo: Max Flow || Maximum Bipartite Matching

#include <bits/stdc++.h>

using namespace std;

#define MAX 40

#define INF 1<<30

int c[ MAX ][ MAX ];

char line[ 30 ], programs[ 15 ];

int prev[ MAX ] , seen[ MAX ];

bool augmentPath( int source , int sink ){

queue< int > Q;

Q.push( source );

memset( seen , 0 , sizeof( seen ) );

seen[ source ] = 1;

prev[ source ] = -1;

int u , v;

while( !Q.empty() ){

u = Q.front(); Q.pop();

if( u == sink ) return true;

for( v = 0 ; v < MAX ; ++v ){

if( c[ u ][ v ] > 0 && !seen[ v ] ){

seen[ v ] = 1;

prev[ v ] = u;

Q.push( v );

}

}

}

return false;

}

int ans[ MAX ];

int maxflow( int source , int sink ){

int flow = 0 , x , mini;

while( augmentPath( source , sink ) ){

mini = INF;

for( x = sink ; prev[ x ] != -1 ; x = prev[ x ] ){

mini = min( mini , c[ prev[ x ] ][ x ] );

}

for( x = sink ; prev[ x ] != -1 ; x = prev[ x ] ){

c[ prev[ x ] ][ x ] -= mini;

c[ x ][ prev[ x ] ] += mini;

}

flow += mini;

}

return flow;

}

int main(){

char computer;

int users , len , source = 38 , sink = 39, i , mf , totUsers , j;

while( gets( line ) ){

sscanf( line , "%c%d %s" , &computer , &users , &programs );

len = strlen( programs );

totUsers = users;

memset( c , 0 , sizeof( c ) );

for( i = 0 ; i < len - 1 ; ++i ){

c[ computer - 'A' + 10 ][ programs[ i ] - '0' ] = 1;

c[ programs[ i ] - '0' ][ sink ] = 1;

}

c[ source ][ computer - 'A' + 10 ] += users;

while( gets( line ) ){

if( strlen( line ) == 0 ) break;

sscanf( line , "%c%d %s" , &computer , &users , &programs );

totUsers += users;

len = strlen( programs );

for( i = 0 ; i < len - 1 ; ++i ){

c[ computer - 'A' + 10 ][ programs[ i ] - '0' ] = 1;

c[ programs[ i ] - '0' ][ sink ] = 1;

}

c[ source ][ computer - 'A' + 10 ] += users;

}

memset( ans , -1 , sizeof( ans ) );

mf = maxflow( source , sink );

if( totUsers != mf ){

puts("!");

continue;

}

for( i = 0 ; i < 10 ; ++i ){

for( j = 10 ; j < 36 ; ++j ){

if( c[ i ][ j ] == 1 ){

break;

}

}

if( j == 36 ) putchar('\_');

else printf("%c" , j - 10 + 'A' );

}

printf("\n");

}

return 0;

}

# Count Cantor

\*\*\*ID: 264

\*\*\*Tipo: Binary Search

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

#define MAX 4775

int val[MAX];

void solve(){

for(int i=1;i<=MAX;i++){

val[i]=i\*(i+1)/2;

}

}

int binarySearch(int a){

int p=1,q=MAX,m;

while(p<=q){

m=(p+q)/2;

if(val[m]==a) {return m;}

if(val[m]<a){

p=m+1;

}

else{

q=m-1;

}

}

return m;

}

int main(){

solve();

int casos,n;

scanf("%d",&casos);

while(casos--){

scanf("%d",&n);

int m=binarySearch(n);

printf("TERM %d IS ",n);

int aux;

if(val[m]<n)aux=m+1;

else aux=m;

if(aux%2==0){

if(n==val[m]){

printf("%d/1",val[m]-val[m-1]);

}

else if(n<val[m]){

printf("%d/%d",n-val[m-1],val[m]-n+1);

}

else if(n>val[m]){

printf("%d/%d",n-val[m],val[m+1]-n+1);

}

}

else{

if(n==val[m]){

printf("1/%d",val[m]-val[m-1]);

}

else if(n<val[m]){

printf("%d/%d",val[m]-n+1,n-val[m-1]);

}

else if(n>val[m]){

printf("%d/%d",val[m+1]-n+1,n-val[m]);

}

}

putchar('\n');

}

return 0;

}

# Simply Syntax

\*\*\*ID: 271

\*\*\*Tipo: Ad hoc Strings

#include <bits/stdc++.h>

using namespace std;

//Gramatica

/\*

S : p|q|r|s|t|u|v|w|x|y|z| NS | CSS | DST | EST | IST

\*/

#define MAX 300

char s[ MAX ];

int correct(){

int i , k;

for( i = 0 , k = 1 ; s[ i ] ; ++i ){

if( s[ i ] == 'C' || s[ i ] == 'D' || s[ i ] == 'E' || s[ i ] == 'I' ){

if( k == 0 ) return 0;

k++;

}

else if( s[ i ] >= 'p' && s[ i ] <= 'z' ){

if( k == 0 ) return 0;

k--;

}

else{

if( k == 0 ) return 0;

}

}

return ( k == 0 );

}

int main(){

while( gets( s ) ){

printf( "%s\n" , (correct() )?"YES":"NO" );

}

return 0;

}

# Chess

\*\*\*ID: 278

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

int min( int x , int y ){ return ( x < y )? x : y; }

int main(){

int t , n , m;

char c;

scanf("%d" , &t );

while( t-- ){

scanf(" %c %d %d" , &c , &m , &n );

if( c == 'k' ) printf("%d\n" ,( ( m \* n ) + 1)/2 ); //((m+1)/2) \* ((n+1)/2) + (m/2) \* (n/2)

else if( c == 'r' ) printf("%d\n" , min( m , n ) );

else if( c == 'Q' ) printf("%d\n" , min( m , n ) );

else printf("%d\n" , ( ( m + 1 )/2 ) \* ( ( n + 1 )/2 ) );

}

return 0;

}

# Vertex

\*\*\*ID: 280

\*\*\*Tipo: Graph Theory, reachability, DFS, BFS

#include <bits/stdc++.h>

using namespace std;

#define MAX 200

vector<int> ady[ MAX ];

int visited[ MAX ];

void dfs( int x ){

for( int i = 0 ; i < ady[ x ].size(); ++i ){

if( !visited[ ady[ x ][ i ] ] ){

visited[ ady[ x ][ i ] ] = 1 ;

dfs( ady[ x ][ i] );

}

}

}

int main(){

int V, n, origen, cont;

bool b;

while( scanf( "%d", &V ) && V ){

while( scanf("%d",&origen) && origen ){

while( scanf("%d", &n ) && n ){

ady[ origen ].push\_back( n );

}

}

scanf("%d", &n );

while( n-- ){

scanf("%d",&origen);

memset( visited, 0, sizeof( visited) );

dfs( origen );

cont = 0;

for( int i = 1; i <= V ;++i ){

if( !visited[ i ] )cont++;

}

printf("%d",cont);

for( int i = 1 ; i <= V; ++i ){

if( !visited[ i ])printf(" %d",i);

}

printf("\n");

}

for( int i = 0 ; i <= V ;++i )ady[ i ].clear();

}

return 0;

}

# The House of Santa Claus

\*\*\*ID: 291

\*\*\*Tipo: Graph Theory, traversal, DFS, Euler

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

char num[10];

char ady[7][7];

void Enlazar() {

ady[1][2] = 1; ady[2][1] = 1;

ady[2][3] = 1; ady[3][2] = 1;

ady[5][2] = 1; ady[2][5] = 1;

ady[3][5] = 1; ady[5][3] = 1;

ady[1][5] = 1; ady[5][1] = 1;

ady[1][3] = 1; ady[3][1] = 1;

ady[5][4] = 1; ady[4][5] = 1;

ady[3][4] = 1; ady[4][3] = 1;

}

void print() {

int i;

printf("1");

for(i = 0; i<8; i ++)

printf("%d",num[i]);

putchar('\n');

}

void dfs(int inicial,int level){

int j,k;

if(level==8){

print();

return;

}

for(int i=1;i<=5;i++){

if(ady[inicial][i] && ady[i][inicial]){

num[level]=i;

ady[i][inicial]=ady[inicial][i]=0;

dfs(i,level+1);

j=level;

k=num[j];

ady[k][inicial]=1;

ady[inicial][k]=1;

}

}

j=level-2;

if(j>=0) {

k = num[j];

ady[k][inicial] = 1;

ady[inicial][k] = 1;

}

}

int main(){

Enlazar();

dfs(1,0);

return 0;

}

# Divisors

\*\*\*ID: 294

\*\*\*Tipo: Number theory, sieve, primes.

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

#define MAX 31630

int prime[MAX];

void sieve(){

for(int i = 2; i <= MAX; ++i){

prime[i]=1;

}

for(int i=2;i\*i<=MAX;i++){

if(prime[i]==1){

for(int k=2;k\*i<=MAX;k++){

prime[k\*i]=0;

}

}

}

}

long getDivisors(long n){

int factores[MAX],id=0;

for(int i=0;i<MAX;i++){

factores[i]=0;

}

long resp=1;

for(int i=2;i\*i<=n;i++){

while(n%i==0 && prime[i]==1){

factores[id]++;

n/=i;

}

if(factores[id]!=0)id++;

}

if(n>1) resp\*=2;

factores[id]=0;

for(int i=0;i<=id;i++){

if(factores[i]!=0)

resp\*=(factores[i]+1);

}

return resp;

}

int main(){

int n,cont,max;

long l,u,i,j,k,q,resp;

sieve();

scanf("%d",&n);

for(i=0;i<n;i++)

{

scanf("%ld %ld",&l,&u);

max=0;

for(j=l;j<=u;j++){

k=getDivisors(j);

if(k>max){max=k;resp=j;}

}

printf("Between %ld and %ld, %ld has a maximum of %ld divisors.\n",l,u,resp,max);

}

return 0;

}

# Safebreaker

\*\*\*ID: 296

\*\*\*Tipo: Brute Force

#include <bits/stdc++.h>

using namespace std;

string toStr(int n){string s;ostringstream buffer;buffer<<n;s=buffer.str();return s;}

string Str( int n ){

string s = toStr( n );

for( int i = s.length() ; i < 4 ; ++i ) s = "0" + s;

return s;

}

#define MAX 15

int main(){

int t, n, valid[ MAX ] , change[ MAX ], cnt, eq , c , j, aux[ MAX ], aux2[ MAX ];

scanf("%d" , &t );

char num[ 15 ][ 5 ];

string ans;

vector<string> v;

for( int i = 0 ; i < 10000 ; ++i ) v.push\_back( Str( i ) );

memset( aux , 0 , sizeof( aux ) );

while( t-- ){

scanf("%d" , &n );

for( int i = 0 ; i < n ; ++i ) scanf("%s %d/%d" , &num[ i ] , &valid[ i ] , &change[ i ] );

cnt = 0;

for( int i = 0 ; i < 10000 ; ++i ){

j = 0;

for( ; j < n ; ++j ){

eq = 0;

memset( aux , 0 , sizeof( aux ) );

memset( aux2 , 0 , sizeof( aux2 ) );

for( int k = 0 ; k < 4 ; ++k ){

if( v[ i ][ k ] == num[ j ][ k ] ) eq++;

else { aux2[ num[ j ][ k ] - '0' ]++; aux[ v[ i ][ k ] - '0' ]++;}

}

if( eq != valid[ j ] ) break;

eq = 0;

for( int k = 0 ; k < 10 ; ++k ){

if( aux2[ k ] && aux[ k ] ) eq += min( aux[ k ] , aux2[ k ] );

}

if( eq != change[ j ] ){ break; }

if( j == n ){ cnt++; ans = v[ i ];}

if( cnt > 1 )break;

}

if( cnt == 2 ) puts("indeterminate");

else if( cnt ) printf("%s\n" , ans.c\_str() );

else puts("impossible");

}

return 0;

}

# Quadtrees

\*\*\*ID: 297

\*\*\*Tipo: recursion

#include <bits/stdc++.h>

using namespace std;

#define MAX 2225

char s1[ MAX ], s2[ MAX ];

int len, sum;

bool seen[ MAX ];

int a[ 6 ][ 1<<17 ], visit[ 6 ][ 1<<17 ];

//marco en a como 1 si tenemos full

/\*

arbol representado como matriz

1 - root

2 - 1 2 3 4

3 - 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 //hijos de nodos de altura 1 (4 \* 40

...

\*/

void solve( int l , string s , int d , int altura , int ant ){

if( l == len ){

return;

}

for( int i = l , j = 0 ; i < len && j < 4; ++i ){

if( seen[ i ] )continue;

if( s[ i ] == 'f' ){

a[ altura ][ ant + j ] = 1;

}

else if( s[ i ] == 'p' ){

solve( i + 1 , s , d - 2 , altura + 1 , ( ant + j ) \* 4 );

}

j++;

seen[ i ] = 1;

}

}

//si al unir dos arboles el nodo padre es negro, todos sus hijos seran empty o nulos

void clean( int i , int j ){

if( i > 5 || j > 1<<16 ){

return;

}

for( int k = 0 ,ini = ( j + k )\*4 ; k < 4 ; ++k ){

a[ i + 1 ][ ini + k ] = 0;

clean( i + 1 , ini + k );

}

}

int main(){

int t;

scanf("%d" , &t );

while( t-- ){

scanf("%s %s" , &s1 , &s2 );

if( s1[ 0 ] == 'f' || s2[ 0 ] == 'f' ){

printf("There are 1024 black pixels.\n");

continue;

}

len = strlen( s1 );

memset( a , 0 , sizeof( a ) );

memset( seen , 0 , sizeof( seen ) );

solve( 1 , s1 , 8 , 1 , 0 );

len = strlen( s2 );

memset( seen , 0 , sizeof( seen ) );

sum = 0;

solve( 1 , s2 , 8 , 1 , 0);

for( int i = 0, aux = 0 , d = 10; i < 6 ; ++i , aux += 2 , d -= 2 ){

for( int j = 0 ; j < 1<<aux ; ++j ){

if( a[ i ][ j ] == 1 ){

sum += 1<<d;

clean( i , j );

}

}

}

printf("There are %d black pixels.\n" , sum );

}

return 0;

}

# Train Swapping

\*\*\*ID: 299

\*\*\*Tipo: Sorting, counting inversions

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

int countBSort(int a[],int l){

int cont=0,i,j,tmp;

for(i=0;i<l-1;i++){

for(j=i+1;j<l;j++){

if(a[j]<a[i]){

std::swap(a[j],a[i]);

cont++;

}

}

}

return cont;

}

int main(){

int n,cantidad,i,resp;

scanf("%d",&n);

while(n--){

scanf("%d",&cantidad);

int array[cantidad];

for(i=0;i<cantidad;i++){

scanf("%d",&array[i]);

}

//BUBLE SORT O(n^2)

resp=countBSort(array,cantidad);

printf("Optimal train swapping takes %d swaps.\n",resp);

}

return 0;

}

# John's Trip

\*\*\*ID: 302

\*\*\*Tipo: Graph Theory, Euler Tour

#include <bits/stdc++.h>

using namespace std;

#define MAX 45

struct Edge{

int v, id;

Edge( int vv, int i ):v(vv), id(i){}

};

vector<Edge> ady[ MAX ];

bool f( Edge e1, Edge e2 ){

return e1.id < e2.id;

}

int V;

#define MAXN 2000

bool seen[ MAXN ];

deque<int> tour;

void dfs( int inicial, int id ){

for( int i = 0 ; i < ady[ inicial ].size(); ++i ){

if( !seen[ ady[ inicial ][ i ].id ] ){

seen[ ady[ inicial ][ i ].id ] = true;

dfs( ady[ inicial ][ i ].v , ady[ inicial ][ i ].id );

}

}

tour.push\_front( id );

}

int main(){

int id, u , v, inicial;

int degree[ MAX ];

while( scanf("%d %d", &u, &v) && u|v ){

memset( degree , 0 , sizeof( degree ) );

scanf("%d", &id );

V = 0;

ady[ u ].push\_back( Edge( v , id ) );

ady[ v ].push\_back( Edge( u , id ) );

degree[ u ]++;

degree[ v ]++;

V = max( V , max( u , v ) );

//por enunciado john vive en la primra calle

inicial = min( u , v );

while( scanf("%d %d", &u, &v) && u|v ){

scanf("%d", &id );

ady[ u ].push\_back( Edge( v , id ) );

ady[ v ].push\_back( Edge( u , id ) );

degree[ u ]++;

degree[ v ]++;

V = max( V , max( u , v ) );

}

bool possible = true;

for( int i = 1 ; i <= V ; ++i ){

if( degree[ i ] & 1 ){

possible = false;

break;

}

}

if( !possible ){

printf("Round trip does not exist.\n\n");

for( int i = 0 ; i <=V ;++i )ady[ i ].clear();

continue;

}

///ordenamos por id de arista para obtener lexicograficamente menor

for( int i = 1 ; i <= V ;++i )sort( ady[ i ].begin() , ady[ i ].end() , f );

memset( seen, 0 , sizeof( seen ) );

tour.clear();

dfs( inicial , - 1);

printf("%d", tour[ 1 ]);

for( int i = 2 ; i < tour.size(); ++i )printf(" %d", tour[ i ] );

printf("\n\n");

for( int i = 0 ; i <=V ;++i )ady[ i ].clear();

}

return 0;

}

# Joseph

\*\*\*ID: 305

\*\*\*Tipo: Ad hoc, Simulation, Joseph problem

#include <bits/stdc++.h>

int t, r[] = {0,2,7,5,30,169,441,1872,7632,1740,93313,459901,1358657,2504881};

int main(){

while (scanf("%d", &t) == 1 && t != 0)

printf("%d\n", r[t]);

return 0;

}

# Packets

\*\*\*ID: 311

\*\*\*Tipo: Greedy

#include <bits/stdc++.h>

int main(){

int a , b , c , d , e , f , ans;

while( scanf("%d %d %d %d %d %d" , &a , &b , &c , &d , &e , &f ) , a | b | c | d |e |f ){

ans = f;

ans += e;

a -= e \* 11;

ans += d;

int falta = d \* ( 36 - 16 );

if( b > 0 ){

int numB = 5 \* d;

if( b >= numB ) b -= numB , falta -= 4 \* numB;

else falta -= 4 \* b , b = 0;

}

if( falta > 0 && a > 0 ){

a -= falta;

}

ans += c/4;

if( c % 4 != 0 ){

falta = ( 36 - 9 \* ( c % 4 ) );

int cubo = c % 4;

if( b > 0 ){

if( cubo == 1 ){

if( b >= 5 ) b -= 5 , falta -= 4 \* 5;

else falta -= 4 \* b , b = 0;

}

else if( cubo == 2 ){

if( b >= 3 ) b -= 3 , falta -= 4 \* 3;

else falta -= 4 \* b , b = 0;

}

else if( cubo == 3 ){

if( b >= 1 ) b -= 1 , falta -= 4;

else falta -= 4 \* b , b = 0;

}

}

if( falta > 0 && a > 0 ){

a -= falta;

}

ans++;

}

ans += b / 9;

if( b % 9 != 0 ){

ans++;

falta = 36 - ( 4 \* (b % 9) );

a -= falta;

}

if( a > 0 ){

ans += a / 36;

if( a % 36 != 0 ) ans++;

}

printf("%d\n" , ans );

}

return 0;

}

# Robot

\*\*\*ID: 314

\*\*\*Tipo: Graph Theory, bfs

#include <bits/stdc++.h>

#include<stdio.h>

#include<cstring>

#include<queue>

using namespace std;

#define MAX 101

struct Estado{

int x; //coordenada x

int y; //coordenada y

int dir; //direccion este,oeste,norte,sur

int d; //distancia

Estado(int x1,int y1,int dir1,int d1):x(x1),y(y1),dir(dir1),d(d1){};

};

int dx[] = { -1, 0, 1, 0};

int dy[] = { 0, 1, 0,-1};

int tabla[MAX][MAX];

void bfs(int f,int c, int x, int y, int xf,int yf,int dir) {

if(x==xf && y==yf){printf("0\n");return;}

if(tabla[x][y]==0 && tabla[xf][yf]==0){

Estado inicial(x,y,dir,0);

queue<Estado> cola;

cola.push(inicial);

int nx,ny;

bool visitado[f][c][4];

memset(visitado,false,sizeof(visitado));

while(!cola.empty()){

Estado actual=cola.front();

cola.pop();

if(tabla[actual.x][actual.y]==1 || visitado[actual.x][actual.y][actual.dir] || actual.x<=0 || actual.y<=0 || actual.x>=f ||actual.y>=c){continue;}

if(actual.x==xf && actual.y==yf){printf("%d\n",actual.d);return;}

visitado[actual.x][actual.y][actual.dir]=true;

Estado izq(actual.x,actual.y,(actual.dir+1)%4,actual.d+1);

Estado der(actual.x,actual.y,(actual.dir+3)%4,actual.d+1);

cola.push(izq);

cola.push(der);

nx=actual.x;

ny=actual.y;

for(int i=0;i<3;i++){

nx+=dx[actual.dir],ny+=dy[actual.dir];//vemos adyacentes

//no tiene caso seguir avanzando si excede limites de matriz

if(nx<=0 || ny<=0 || nx>=f || ny>=c || tabla[nx][ny]==1)break;

Estado ady(nx,ny,actual.dir,actual.d+1); //colocamos a la cola los pasos que puede avanzar

cola.push(ady);

}

}

}

printf("-1\n");

}

int main(){

int f,c,j,i,xi,yi,xf,yf,direccion;

char estado[5];

while(scanf("%d %d",&f,&c) && f+c){

int a[f][c];

for(i=0;i<f;i++){

for(j=0;j<c;j++){

scanf("%d",&a[i][j]);

}

}

memset(tabla,0,sizeof(tabla));

//modifico arreglo ingresado

for(i=0;i<f;i++){

for(j=0;j<c;j++){

if(a[i][j]){

tabla[i][j]=1;

tabla[i][j+1]=1;

tabla[i+1][j]=1;

tabla[i+1][j+1]=1;

}

}

}

scanf("%d %d %d %d %s",&xi,&yi,&xf,&yf,&estado);

switch(estado[0]){

case 'n': direccion=0;break;

case 'e': direccion=1;break;

case 's': direccion=2;break;

case 'w': direccion=3;break;

}

bfs(f,c,xi,yi,xf,yf,direccion);

}

return 0;

}

# Network

\*\*\*ID: 315

\*\*\*Tipo: Graph Theory, Articulation Point, DFS

///PUNTO ARTICULACION

///Si raiz tiene mas de un hijo es punto de articulacion

///Recuerde acerca de aristas de arbol(padre-hijo) y aristas de retroceso(antecesor-descendiente)

///low[k] posee el numero de dfs mas bajo a partir de un nodo, si se ve como arbol seria el mas alto

#include <cstdlib>

#include <cstdio>

#include <iostream>

#include<sstream>

#include<cstring>

#include<vector>

using namespace std;

#define MAXN 110

#define min(a,b) ( (a)<(b)?(a):(b) )

vector<int> ady[MAXN];

int n, cnt;

bool visitado[MAXN], flag[MAXN];

int dfsNum[MAXN], low[MAXN];

void dfs(int k)

{

++cnt;

dfsNum[k]=cnt,low[k]=cnt,visitado[k]= true;

int hijos= 0;

//adyacentes de la lista

for( size\_t i=0; i<ady[k].size(); ++i )

{

if(!visitado[ady[k][i]])

{

hijos++;

dfs(ady[k][i]);

low[k]=min(low[k],low[ady[k][i]]);

//si no es raiz y el numero de dfs del adyacente(low) es igual al numeor de dfs del actual es punto de articulacion

if( k!=1 && low[ady[k][i]]==dfsNum[k])flag[k]=true;

//si es raiz y tiene mas de un hijo es puntoarticulacion

if( k==1 && hijos>1 ) flag[k]=true;

}

else if(k!=ady[k][i]){

///el mas bajo sera el minimo entre el mas bajo actual y el numero de antecesor

low[k]= min(low[k],dfsNum[ady[k][i]]);

}

}

}

void Init(){

for( int i= 0; i<= MAXN; ++i )

ady[i].clear();

memset( visitado, false, sizeof(visitado) );

memset( flag, false, sizeof(flag) );

memset( dfsNum, 0, sizeof(flag) );

memset( low, 0, sizeof(low) );

}

int main(void)

{

int x,y,n;

while(scanf("%d",&n) && n!=0){

Init();

cin.get();

while(1){

string s;

getline(cin,s);

s+=" ";

stringstream ss(s);

ss>>x;

if(x==0)break;

while(ss>>y){

ady[x].push\_back(y);

ady[y].push\_back(x);

}

}

cnt=0;

dfs(1);

int num= 0;

for( int i= 1; i<= n; ++i )

if( flag[i] ){ num++;}

printf("%d\n", num );

}

return 0;

}

# The New Villa

\*\*\*ID: 321

\*\*\*Tipo: BFS + Bitmasks

#include <bits/stdc++.h>

using namespace std;

#define MAX 15

vector< int > Switch[ MAX ] , ady[ MAX ];

int V;

struct Previo{

int x , mask , id;

string command;

Previo( int xx , int m , string c , int i ): x( xx ) , mask( m ) , command( c ) , id( i ){}

Previo(){}

}prev[ MAX ][ 1<<11 ];

bool seen[ MAX ][ 1<<11 ]; //cuarto en el que estoy y las luces encendidas hasta el momento

struct Estado{

int x , d , mask;

Estado( int xx , int dd , int m ): x( xx ) , d( dd ), mask( m ){}

};

void print( int x , int mask ){

if( prev[ x ][ mask ].x == -1 ){

return;

}

print( prev[ x ][ mask ].x , prev[ x ][ mask ].mask );

printf("- %s %d.\n" , prev[ x ][ mask ].command.c\_str() , prev[ x ][ mask ].id );

}

void bfs(){

queue< Estado > Q;

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

int v, ans = 1<<30;

prev[ 0 ][ 1 ] = Previo( -1 , -1 , "" , -1 );

memset( seen , 0 ,sizeof( seen ) );

seen[ 0 ][ 1 ] = 1;

while( !Q.empty() ){

Estado act = Q.front(); Q.pop();

if( act.x == V - 1 && act.mask == 1<<( V - 1 ) ){ //si estoy en el cuarto final con las demas luces apagadas

ans = act.d;

break;

}

int mask = act.mask;

//presiono switch en el room actual

for( int i = 0 ; i < Switch[ act.x ].size() ; ++i ){

v = Switch[ act.x ][ i ];

if( v == act.x ) continue;

mask = act.mask;

if( mask & 1<<v ){ //si esta encendido el cuarto actual, lo apago

mask = mask & ( ~( 1<<v ) );

if( !seen[ act.x ][ mask ] ){

Q.push( Estado( act.x , act.d + 1 , mask ) );

seen[ act.x ][ mask ] = 1;

prev[ act.x ][ mask ] = Previo( act.x , act.mask , "Switch off light in room" , v + 1 );

}

}

else{

mask |= 1<<v; //sino lo enciendo

if( !seen[ act.x ][ mask ] ){

Q.push( Estado( act.x , act.d + 1 , mask ) );

seen[ act.x ][ mask ] = 1;

prev[ act.x ][ mask ] = Previo( act.x , act.mask , "Switch on light in room" , v + 1 );

}

}

}

//me muevo a otro room siempre y cuando este la luz encendida

for( int i = 0 ; i < ady[ act.x ].size() ; ++i ){

v = ady[ act.x ][ i ];

if( ( act.mask & 1<<v ) && !seen[ v ][ act.mask ] ){

Q.push( Estado( v , act.d + 1 , act.mask ) );

seen[ v ][ act.mask ] = 1;

prev[ v ][ act.mask ] = Previo( act.x , act.mask , "Move to room" , v + 1 );

}

}

}

if( ans == 1<<30 ){

puts("The problem cannot be solved.");

}

else{

printf("The problem can be solved in %d steps:\n" , ans );

print( V - 1 , 1<<( V - 1 ) ); //imprimo recursivamente

}

}

int main(){

int u , v , E , S , q = 1;

while( scanf("%d %d %d" , &V , &E , &S ) , V | E | S ){

for( int i = 0 ; i <= V ; ++i ){

ady[ i ].clear();

Switch[ i ].clear();

}

while( E-- ){

scanf("%d %d" , &u , &v );

u--; v--;

ady[ u ].push\_back( v );

ady[ v ].push\_back( u );

}

while( S-- )

{

scanf("%d %d" , &u , &v );

u--;v--;

Switch[ u ].push\_back( v );

}

printf("Villa #%d\n" , q++ );

bfs();

printf("\n");

}

return 0;

}

# Factorial Frequencies

\*\*\*ID: 324

\*\*\*Tipo: Ad hoc, bignum.

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

const int MAXD = 785, DIG = 9, BASE = 1000000000;

const unsigned long long BOUND = numeric\_limits <unsigned long long> :: max () - (unsigned long long) BASE \* BASE;

struct bignum

{

int D, digits [MAXD / DIG + 2];

inline void trim ()

{

while (D > 1 && digits [D - 1] == 0)

D--;

}

inline void init (long long x)

{

memset (digits, 0, sizeof (digits));

D = 0;

do

{

digits [D++] = x % BASE;

x /= BASE;

}

while (x > 0);

}

inline bignum (long long x)

{

init (x);

}

inline bignum (int x = 0)

{

init (x);

}

inline bignum (char \*s)

{

memset (digits, 0, sizeof (digits));

int len = strlen (s), first = (len + DIG - 1) % DIG + 1;

D = (len + DIG - 1) / DIG;

for (int i = 0; i < first; i++)

digits [D - 1] = digits [D - 1] \* 10 + s [i] - '0';

for (int i = first, d = D - 2; i < len; i += DIG, d--)

for (int j = i; j < i + DIG; j++)

digits [d] = digits [d] \* 10 + s [j] - '0';

trim ();

}

inline char \*str ()

{

trim ();

char \*buf = new char [DIG \* D + 1];

int pos = 0, d = digits [D - 1];

do

{

buf [pos++] = d % 10 + '0';

d /= 10;

}

while (d > 0);

reverse (buf, buf + pos);

for (int i = D - 2; i >= 0; i--, pos += DIG)

for (int j = DIG - 1, t = digits [i]; j >= 0; j--)

{

buf [pos + j] = t % 10 + '0';

t /= 10;

}

buf [pos] = '\0';

return buf;

}

inline bool operator < (const bignum &o) const

{

if (D != o.D)

return D < o.D;

for (int i = D - 1; i >= 0; i--)

if (digits [i] != o.digits [i])

return digits [i] < o.digits [i];

return false;

}

inline bool operator == (const bignum &o) const

{

if (D != o.D)

return false;

for (int i = 0; i < D; i++)

if (digits [i] != o.digits [i])

return false;

return true;

}

inline bignum operator << (int p) const

{

bignum temp;

temp.D = D + p;

for (int i = 0; i < D; i++)

temp.digits [i + p] = digits [i];

for (int i = 0; i < p; i++)

temp.digits [i] = 0;

return temp;

}

inline bignum operator >> (int p) const

{

bignum temp;

temp.D = D - p;

for (int i = 0; i < D - p; i++)

temp.digits [i] = digits [i + p];

for (int i = D - p; i < D; i++)

temp.digits [i] = 0;

return temp;

}

inline bignum range (int a, int b) const

{

bignum temp = 0;

temp.D = b - a;

for (int i = 0; i < temp.D; i++)

temp.digits [i] = digits [i + a];

return temp;

}

inline bignum operator + (const bignum &o) const

{

bignum sum = o;

int carry = 0;

for (sum.D = 0; sum.D < D || carry > 0; sum.D++)

{

sum.digits [sum.D] += (sum.D < D ? digits [sum.D] : 0) + carry;

if (sum.digits [sum.D] >= BASE)

{

sum.digits [sum.D] -= BASE;

carry = 1;

}

else

carry = 0;

}

sum.D = max (sum.D, o.D);

sum.trim ();

return sum;

}

inline bignum operator - (const bignum &o) const

{

bignum diff = \*this;

for (int i = 0, carry = 0; i < o.D || carry > 0; i++)

{

diff.digits [i] -= (i < o.D ? o.digits [i] : 0) + carry;

if (diff.digits [i] < 0)

{

diff.digits [i] += BASE;

carry = 1;

}

else

carry = 0;

}

diff.trim ();

return diff;

}

inline bignum operator \* (const bignum &o) const

{

bignum prod = 0;

unsigned long long sum = 0, carry = 0;

for (prod.D = 0; prod.D < D + o.D - 1 || carry > 0; prod.D++)

{

sum = carry % BASE;

carry /= BASE;

for (int j = max (prod.D - o.D + 1, 0); j <= min (D - 1, prod.D); j++)

{

sum += (unsigned long long) digits [j] \* o.digits [prod.D - j];

if (sum >= BOUND)

{

carry += sum / BASE;

sum %= BASE;

}

}

carry += sum / BASE;

prod.digits [prod.D] = sum % BASE;

}

prod.trim ();

return prod;

}

inline double double\_div (const bignum &o) const

{

double val = 0, oval = 0;

int num = 0, onum = 0;

for (int i = D - 1; i >= max (D - 3, 0); i--, num++)

val = val \* BASE + digits [i];

for (int i = o.D - 1; i >= max (o.D - 3, 0); i--, onum++)

oval = oval \* BASE + o.digits [i];

return val / oval \* (D - num > o.D - onum ? BASE : 1);

}

inline pair <bignum, bignum> divmod (const bignum &o) const

{

bignum quot = 0, rem = \*this, temp;

for (int i = D - o.D; i >= 0; i--)

{

temp = rem.range (i, rem.D);

int div = (int) temp.double\_div (o);

bignum mult = o \* div;

while (div > 0 && temp < mult)

{

mult = mult - o;

div--;

}

while (div + 1 < BASE && !(temp < mult + o))

{

mult = mult + o;

div++;

}

rem = rem - (o \* div << i);

if (div > 0)

{

quot.digits [i] = div;

quot.D = max (quot.D, i + 1);

}

}

quot.trim ();

rem.trim ();

return make\_pair (quot, rem);

}

inline bignum operator / (const bignum &o) const

{

return divmod (o).first;

}

inline bignum operator % (const bignum &o) const

{

return divmod (o).second;

}

inline bignum power (int exp) const

{

bignum p = 1, temp = \*this;

while (exp > 0)

{

if (exp & 1) p = p \* temp;

if (exp > 1) temp = temp \* temp;

exp >>= 1;

}

return p;

}

};

#define MAX 367

bignum dp[MAX];

void init(){

dp[1]=1;

bignum aux;

for(int i=2;i<MAX;i++){

dp[i]=dp[i-1]\*i;

}

}

int main(){

init();

int n,i;

int digits[10];

bignum aux;

while(scanf("%d",&n) && n!=0){

for(i=0;i<10;i++)digits[i]=0;

aux=dp[n];

while(bignum(0)<aux){

string s=(aux%10).str();

digits[(s[0]-'0')]++;

aux=aux/10;

}

printf("%d! --\n",n);

for(i=0;i<10;i++){

if(i==5)putchar('\n');

if(i==0 || i==5)printf(" (%d)%5d",i,digits[i]);

else printf(" (%d)%5d",i,digits[i]);

}

putchar('\n');

}

return 0;

}

# Mapping the Swaps

\*\*\*ID: 331

\*\*\*Tipo: BFS

#include <bits/stdc++.h>

using namespace std;

#define MAX 10

int n, a[ MAX ];

int cases;

#define psi pair< string , int >

void bfs( string ini , string finish ){

queue< psi > Q;

Q.push( psi( ini , 0 ) );

string s;

int dist[ 1005 ]; memset( dist , 0 , sizeof( dist ) );

bool d[ 1005 ] = { 0 };

if( ini == finish ){

printf("There are %d swap maps for input data set %d.\n" , 0 , cases++);

return;

}

while( !Q.empty() ){

psi act = Q.front(); Q.pop();

if( act.first == finish ){

dist[ act.second ]++;

d[ act.second ] = 1;

continue;

}

for( int i = 0 ; i < n - 1 ; ++i ){

s = act.first;

if( a[ s[ i ] - '0' ] > a[ s[ i + 1 ] - '0' ] ){

swap( s[ i ] , s[ i + 1 ] );

Q.push( psi( s , act.second + 1 ) );

}

}

}

int mini = 1<<30, ans = 0;

for( int i = 0 ; i < 1005 ; ++i ){

if( d[ i ] ){

ans = dist[ i ];

break;

}

}

printf("There are %d swap maps for input data set %d.\n" , ans , cases++);

}

#define pii pair<int ,int >

bool cmp( pii p1 , pii p2 ){

return p1.first < p2.first;

}

int main(){

string ini , finish;

vector< pii > v;

cases = 1;

while( scanf("%d" , &n ) , n ){

ini = "";

v.clear();

for( int i = 0 ; i < n ; ++i ){

scanf("%d" , &a[ i ] );

v.push\_back( pii( a[ i ] , i ) );

ini += (char)( i + '0' );

}

sort( v.begin() , v.end() , cmp );

finish = "";

for( int i = 0 ; i < n ; ++i ) finish += (char)( v[ i ].second + '0' );

bfs( ini , finish );

}

return 0;

}

# Identifying Concurrent Events

\*\*\*ID: 334

\*\*\*Tipo: Graph Theory, Floyd Transitive Closure

#include <bits/stdc++.h>

#define MAX 305

int ady[ MAX ][ MAX ] , V;

void TransitiveClosure(){

for( int k = 0 ; k < V ; ++k ){

for( int i = 0 ; i < V ; ++i ){

for( int j = 0 ; j < V ; ++j ){

ady[ i ][ j ] |= ady[ i ][ k ] & ady[ k ][ j ];

}

}

}

}

char cities[ MAX ][ 40 ];

int numcities;

int get\_Id( char \*s ){

for( int i = 0 ; i < numcities ; ++i ){

if( strcmp( s , cities[ i ] ) == 0 ) return i;

}

strcpy( cities[ numcities ] ,s );

return numcities++;

}

char \*get\_String( int id ){

for( int i = 0 ; i < numcities ; ++i ){

if( id == i ) return cities[ i ];

}

}

int main(){

int q = 1, n , E ,ant , id;

char s[ 40 ] , s1[ 40 ];

while( scanf("%d" , &n ) && n ){

numcities = 0;

while( n-- ){

scanf("%d" , &E );

for( int i = 0 ; i < E ; ++i ){

scanf("%s" , &s );

id = get\_Id( s );

if( i != 0 ){

ady[ ant ][ id ] = 1;

}

ant = id;

}

}

V = numcities;

scanf("%d" , &E );

while( E-- ){

scanf("%s %s" , &s , &s1 );

ady[ get\_Id( s ) ][ get\_Id( s1 ) ] = 1;

}

TransitiveClosure();

int total\_events = V \* ( V - 1 ) / 2;

int x[ 2 ] , y[ 2 ] , idx = 0;

for( int i = 0 ; i < V ; ++i ){

for( int j = i + 1 ; j < V ; ++j ){

if( ady[ i ][ j ] == 1 ||ady[ j ][ i ] == 1 ) total\_events--;

else{

if( idx < 2 ){

x[ idx ] = i;

y[ idx++ ] = j;

}

}

}

}

printf("Case %d, ", q++ );

if( total\_events == 0 )printf("no concurrent events.\n");

else if( total\_events == 1 ){

printf("1 concurrent events:\n");

printf("(%s,%s) \n" , get\_String( x[ 0 ] ) , get\_String( y[ 0 ] ) );

}

else{

printf("%d concurrent events:\n" , total\_events );

printf("(%s,%s) (%s,%s) \n" , get\_String( x[ 0 ] ) , get\_String( y[ 0 ] ) , get\_String( x[ 1 ] ) , get\_String( y[ 1 ] ) );

}

memset( ady , 0 ,sizeof( ady ) );

}

return 0;

}

# A Node Too Far

\*\*\*ID: 336

\*\*\*Tipo: Graph theory, bfs, floyd warshall

#include <bits/stdc++.h>

#include<stdio.h>

#include<vector>

#include<queue>

#include<map>

#include<set>

using namespace std;

#define MAX 55

int ady[MAX][MAX];

struct Estado{

int x;

int d;

Estado(int x1,int d1):x(x1),d(d1){};

};

int bfs(int x,int dist,int tamano){

Estado inicial (x,0);

queue<Estado> cola;

cola.push(inicial);

bool visitado[MAX];

memset(visitado,false,sizeof(visitado));

int cantidad=0;

set<int> resp;

visitado[x]=true;

while(!cola.empty()){

Estado actual=cola.front();

cola.pop();

if(actual.d<=dist){

resp.insert(actual.x);

}

for(int j=0;j<MAX;j++){

if(ady[actual.x][j]==1 && !visitado[j]){

Estado ady(j,actual.d+1);

visitado[j]=true;

cola.push(ady);

}

}

}

return resp.size();

}

int main(){

int cant,i,j,orig,dest,k,dist,nvisit,casos=1;

while(scanf("%d",&cant) && cant!=0){

k=1;

map<int,int> mp;

memset(ady,0,sizeof(ady));

for(i=0;i<cant;i++){

scanf("%d %d",&orig,&dest);

//mapeo valores grandes en pequeños ejem nodo 35 sera nodo 1

if(mp.find(orig)==mp.end()){

mp[orig]=k++;

}

if(mp.find(dest)==mp.end()){

mp[dest]=k++;

}

ady[mp[orig]][mp[dest]]=ady[mp[dest]][mp[orig]]=1; //nodos a matriz

}

while(scanf("%d %d",&orig,&dist)){

if(orig==0 && dist==0)break;

nvisit=k-1-bfs(mp[orig],dist,k);

printf("Case %d: %d nodes not reachable from node %d with TTL = %d.\n",casos++,nvisit,orig,dist);

}

}

return 0;

}

# Packets

\*\*\*ID: 341

\*\*\*Tipo: Graph theory, Shortest paths, SSSP.

#include <bits/stdc++.h>

using namespace std;

#define MAX 15

#define INF (1<<20)

#define pii pair< int, int >

#define pb push\_back

struct comp {

bool operator() (const pii &a, const pii &b) {

return a.second > b.second;

}

};

int D[MAX];

bool F[MAX];

int parent[MAX];

int main() {

int i, u, v, w, sz, nodes, edges, starting,fin,cont=1;

// creamos grafo

while(scanf("%d", &nodes) && nodes){

priority\_queue< pii, vector< pii >, comp > Q;

vector< pii > G[MAX];

for(i=1;i<=nodes;i++){

scanf("%d",&edges);

while(edges--){

scanf("%d %d",&v,&w);

G[i].pb(pii(v,w));

}

}

scanf("%d %d", &starting,&fin);

// iniciamos grafo

for(i=1; i<=nodes; i++) {D[i] = INF;parent[i]=-1;F[i]=0;}

D[starting] = 0;

Q.push(pii(starting, 0));

// dijkstra

while(!Q.empty()) {

u = Q.top().first;

Q.pop();

if(F[u]) continue;

sz = G[u].size();

for(i=0; i<sz; i++) {

v = G[u][i].first;

w = G[u][i].second;

if(!F[v] && D[u]+w < D[v]) {

D[v] = D[u] + w;

parent[v]=u; // ruta mas corta

Q.push(pii(v, D[v]));

}

}

F[u] = 1; // visitado u

}

printf("Case %d: Path =",cont++);

int ruta[100],tam=0;

int end=fin;

ruta[tam++]=fin;

while(parent[end]!=-1){

ruta[tam++]=parent[end];

end=parent[end];

}

for(i=tam-1;i>=0;i--){

printf(" %d",ruta[i]);

}

printf("; %d second delay\n",D[fin]);

}

return 0;

}

# What Base Is This?

\*\*\*ID: 343

\*\*\*Tipo: Number Theory

#include <bits/stdc++.h>

#include<stdio.h>

#include<math.h>

#include<stdlib.h>

#include<string>

using namespace std;

int valor(char n){

int j,i;

for(i='A',j=10;i<='Z' && j<=36;i++,j++){

if(n==i)break;

}

return j;

}

int pow(int a,int b){

int r=1;

for(int i=0;i<b;i++){

r\*=a;

}

return r;

}

int base(string n,int b){

int i,idx=0;

int r=0,aux;

for(i=n.length()-1;i>=0;i--){

if(isdigit(n[i])){

aux=n[i]-'0';

}

else{

aux=valor(n[i]);

}

if(aux>=b)return -1;

r+=(aux\*(int)pow(b,idx));

idx++;

}

return r;

}

int main(){

int i,j;

string x,y;

int bx,by;

while(cin>>x>>y){

for(i=2;i<=36;i++){

bx=base(x,i);

if(bx==-1)continue;

for(j=2;j<=36;j++){

by=base(y,j);

if(by==-1)continue;

if(bx==by)break;

}

if(bx==by)break;

}

if(i==37 || j==37) cout<<x<<" is not equal to "<<y<<" in any base 2..36\n";

else cout<<x<<" (base "<<i<<") = "<<y<<" (base "<<j<<")\n";

}

return 0;

}

# Roman Digititis

\*\*\*ID: 344

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

int tabla[102][6];

string roman[102];

void romanos(){

roman[0]="";

roman[1]="i";

roman[2]="ii";

roman[3]="iii";

roman[4]="iv";

roman[5]="v";

roman[6]="vi";

roman[7]="vii";

roman[8]="viii";

roman[9]="ix";

roman[10]="x";

roman[20]="xx";

roman[30]="xxx";

roman[40]="xl";

roman[50]="l";

roman[60]="lx";

roman[70]="lxx";

roman[80]="lxxx";

roman[90]="xc";

roman[100]="c";

int aux;

for(int i=11;i<100;i++){

if(i<20){aux=10;}

else if(i>=20 && i<30){aux=20;}

else if(i>=30 && i<40){aux=30;}

else if(i>=40 && i<50){aux=40;}

else if(i>=50 && i<60){aux=50;}

else if(i>=60 && i<70){aux=60;}

else if(i>=70 && i<80){aux=70;}

else if(i>=80 && i<90){aux=80;}

else if(i>=90 && i<100){aux=90;}

roman[i]=roman[aux]+roman[i-aux];

}

}

void cantidades(){

int i,j,l,conti=0,contv=0,contx=0,contl=0,contc=0;

for(i=1;i<=100;i++){

l=roman[i].length();

for(j=0;j<l;j++){

if(roman[i][j]=='i'){conti++;}

else if(roman[i][j]=='v'){contv++;}

else if(roman[i][j]=='x'){contx++;}

else if(roman[i][j]=='l'){contl++;}

else if(roman[i][j]=='c'){contc++;}

}

tabla[i][0]=conti;

tabla[i][1]=contv;

tabla[i][2]=contx;

tabla[i][3]=contl;

tabla[i][4]=contc;

}

}

int main(){

int l,i,n;

romanos();

cantidades();

while(scanf("%d",&n) && n!=0){

printf("%d: %d i, %d v, %d x, %d l, %d c\n",n,tabla[n][0],tabla[n][1],tabla[n][2],tabla[n][3],tabla[n][4]);

}

return 0;

}

# Pseudo-Random Numbers

\*\*\*ID: 350

\*\*\*Tipo: Cycle Finding

#include <bits/stdc++.h>

int Z, I , M, L;

int f( int L ){

return ( Z \* L + I ) % M;

}

void TortoiseHare(int x)

{

int s = 0, p = 1;

// i es la tortuga, para iniciar el siguiente paso, j es el conejo, hasta el punto de partida para los dos pasos siguientes.

int i = f(x), j = f(f(x));

// Velocidad de una tortuga, conejo de velocidad doble. Trate de encontrar un ciclo de azar.

while (i != j) i = f(i), j = f(f(j));

// Tortugas hasta el punto de partida en el conejo a la tortuga original.

j = i; i = x;

// La tortuga y la liebre son uno de velocidad. Comience a buscar bucle punto de partida.

while (i != j) i = f(i), j = f(j), s++;

// Los conejos de las tortugas el siguiente paso.

j = f(i);

// Tortuga no se mueve, una velocidad de conejo. Comience a buscar por el período mínimo.

while (i != j) j = f(j), p++;

printf("%d\n",p);

}

int main(){

int t = 1;

while( scanf("%d %d %d %d", &Z, &I, &M, &L) && (Z + I + M + L ) ){

printf("Case %d: ", t++ );

TortoiseHare( L );

}

return 0;

}

# Seasonal War

\*\*\*ID: 352

\*\*\*Tipo: Graph Theory, BFS, DFS

#include <bits/stdc++.h>

#include<stdio.h>

#include<queue>

#include<vector>

#include<algorithm>

using namespace std;

#define MAX 30

char ady[MAX][MAX];

bool visitado[MAX][MAX];

int h,w;

struct Estado{

int x,y;

Estado(int x1,int y1):x(x1),y(y1){}

};

int dx[8]={1,-1,0, 0,1,-1,-1,1};

int dy[8]={0, 0,1,-1,1, 1,-1,-1};

void bfs(int x, int y, char c){

Estado inicial(x,y);

queue<Estado> Q;

Q.push(inicial);

while(!Q.empty()){

Estado actual=Q.front();Q.pop();

visitado[actual.x][actual.y]=true;

for(int i=0;i<8;i++){

int nx=dx[i]+actual.x;

int ny=dy[i]+actual.y;

if(nx>=0 && nx<h && ny>=0 && ny<w && ady[nx][ny]==c && !visitado[nx][ny]){

Estado vecino(nx,ny);

Q.push(vecino);

}

}

}

return ;

}

int main(){

int resp;

string s;

int ii=1;

while(scanf("%d",&h)!=EOF){

w=h;

for(int i=0;i<h;i++){

for(int j=0;j<w;j++){

ady[i][j]=' ';

visitado[i][j]=false;

}

}

for(int i=0;i<h;i++){

cin>>s;

for(int j=0;j<w;j++){

ady[i][j]=s[j];

}

}

resp=0;

for(int i=0;i<h;i++){

for(int j=0;j<w;j++){

if(!visitado[i][j] && ady[i][j]=='1'){

bfs(i,j,'1');

resp++;

}

}

}

printf("Image number %d contains %d war eagles.\n",ii++,resp);

}

return 0;

}

# Pesly Palindromes

\*\*\*ID: 353

\*\*\*Tipo: Ad hoc, String matching

#include <iostream>

#include <map>

using namespace std;

bool isPalindrome(string c){

int l = c.length();

for(int i=0,j=l-1;i<=l/2 && j>=l/2;i++,j--){

if(c[i]!=c[j])return false;

}

return true;

}

string substring( string s , int ini , int fin ){

string ss="";

for( int i = ini; i<= fin ; ++i ){

ss+=s[i];

}

return ss;

}

int main(){

string s , sub;

int l, cont;

while( cin>>s ){

map<string ,bool > mp;

l = s.length();

cont = 0;

for( int i = 0 ; i < l ; ++i){

for( int j = i ; j < l ; ++j ){

sub = substring(s , i , j);

if( sub.length() > 0 && !mp[ sub ] && isPalindrome( sub ) ){ mp[ sub] = true; cont++;}

}

}

cout<<"The string '"<<s<<"' contains "<<cont<<" palindromes."<<endl;

}

return 0;

}

# The Bases are Loaded

\*\*\*ID: 355

\*\*\*Tipo: Number Theory

#include <bits/stdc++.h>

#include<stdio.h>

#include<string>

#include<sstream>

using namespace std;

int valor(char n){

int j,i;

for(i='A',j=10;i<='F' && j<=15;i++,j++){

if(n==i)break;

}

return j;

}

char valor2(int n){

int i,j;

for(i=10,j='A';i<=15 && j<='F';i++,j++){

if(n==i)break;

}

return (char)j;

}

long long pow(long long a,long long b){

long long r=1;

for(int i=0;i<b;i++){

r\*=a;

}

return r;

}

long long base(string n,int b){

int i,idx=0,aux;

long long r=0;

for(i=n.length()-1;i>=0;i--){

if(isdigit(n[i])){

aux=n[i]-'0';

}

else{

aux=valor(n[i]);

}

if(aux>=b)return -1;

r+=(aux\*pow(b,idx));

idx++;

}

return r;

}

string toString(long long n){

string s;

ostringstream buffer;

buffer<<n;

s= buffer.str();

return s;

}

string base2(long long n,long long b){

string str="";

int aux;

if(n==0)return "0";

while(n>=1){

aux=n%b;

if(aux>9 && aux<16){str=valor2(aux)+str;}

else {str=toString(aux)+str;}

n/=b;

}

return str;

}

int main(){

string n;

long long bi,bf,d,resp;

while(cin>>bi>>bf>>n){

d=base(n,bi);

if(d==-1)cout<<n<<" is an illegal base "<<bi<<" number"<<endl;

else{

if(bf==10)cout<<n<<" base "<<bi<<" = "<<d<<" base "<<bf<<endl;

else{

cout<<n<<" base "<<bi<<" = "<<base2(d,bf)<<" base "<<bf<<endl;

}

}

}

return 0;

}

# Let Me Count The Ways

\*\*\*ID: 357

\*\*\*Tipo: DP, Coin Change

#include <bits/stdc++.h>

#define MAX 30005

long long dp[ MAX ];

int a[ 5 ] = { 1 , 5 , 10 , 25 , 50 };

void CoinChange(){

dp[ 0 ] = 1;

for( int i = 0 ; i < 5 ; ++i )

for( int j = a[ i ] ; j < MAX ; ++j )

dp[ j ] += dp[ j - a[ i ] ];

}

int main(){

CoinChange();

int n;

while( scanf("%d" , &n ) != EOF ){

if( dp[ n ] == 1 ) printf("There is only 1 way to produce %d cents change.\n" , n );

else printf("There are %lld ways to produce %d cents change.\n" , dp[ n ] , n );

}

return 0;

}

# Combinations

\*\*\*ID: 369

\*\*\*Tipo: Number theory, Binomial theorem.

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

long long Binomial(long n,long m){

long long resp;

resp=1;

if(m>n/2)m=n-m;

for(int i=0;i<m;i++){

resp\*=n--;

resp/=i+1;

}

return resp;

}

int main(){

long n,m;

while(cin>>n>>m){

if(n==0 && m==0)break;

cout<<n<<" things taken "<<m<<" at a time is "<<Binomial(n,m)<<" exactly."<<endl;

}

return 0;

}

# WhatFix Notation

\*\*\*ID: 372

\*\*\*Tipo: Recursion

#include <bits/stdc++.h>

using namespace std;

#define MAX 1005

char pre[ MAX ] , in[ MAX ];

void Solve( char \*pre , char \*in, int l){

if( l < 1) return;

if( l == 1 ){

putchar( pre[ 0 ] );

return;

}

int i;

for( i = 0 ; i < l ; ++i ){

if( pre[ 0 ] == in[ i ] )break;

}

Solve( pre + 1 , in , i );

Solve( pre + i + 1 , in + i + 1 , l - i - 1);

putchar( pre[ 0 ] );

}

int main(){

while( gets( in ) ){

gets( pre );

printf("INFIX => %s\n" , in );

printf("PREFIX => %s\n" , pre );

printf("POSTFIX => ");

Solve( pre , in , strlen( in ) );

printf("\n");

}

return 0;

}

# Cowculation

\*\*\*ID: 377

\*\*\*Tipo: Number Theory, Bases(base - 4)

#include <bits/stdc++.h>

#include<stdio.h>

#include<sstream>

using namespace std;

string completar(string str,string num2){

for(int i=0;i<8-num2.length();i++){

str="0"+str;

}

return str+=num2;

}

string completar2(string str,string num2){

for(int i=0;i<num2.length()-str.length();i++){

str="0"+str;

}

return str;

}

string convertir(string n){

string resp="";

for(int i=0;i<n.length();i++){

if(n[i]=='V')resp+="0";

if(n[i]=='U')resp+="1";

if(n[i]=='C')resp+="2";

if(n[i]=='D')resp+="3";

}

return resp;

}

string toString(int n){

string r;

stringstream s;

s<<n;

s>>r;

return r;

}

string simboloA(string n1,string n2){

string resp="";

if(n1.length()<n2.length()){n1=completar2(n1,n2);}

if(n2.length()<n1.length()){n2=completar2(n2,n1);}

int carry=0,c1,c2;

for(int i=n1.length()-1;i>=0;i--){

c1=n1[i]-'0';

c2=n2[i]-'0';

if(c1+c2+carry>=4){

resp=toString(c1+c2+carry-4)+resp;

carry=1;

}

else {resp=toString(c1+c2+carry)+resp;carry=0;}

}

if(carry>0)resp=toString(carry)+resp;

return resp;

}

string simboloL(string n2){

return n2+"0";

}

string simboloR(string n2){

return "0"+n2.substr(0,n2.length()-1);

}

string simboloN(string n2){

return n2;

}

string opcion(char s,string n1,string n2){

string resp;

switch(s){

case 'A': resp=simboloA(n1,n2);

break;

case 'L': resp=simboloL(n2);

break;

case 'R': resp=simboloR(n2);

break;

case 'N': resp=simboloN(n2);

break;

}

return resp;

}

int main(){

int n,i,;

string num1,num2,resp,str;

char s1,s2,s3;

scanf("%d",&n);

for(i=0;i<n;i++){

cin>>num1>>num2>>s1>>s2>>s3>>resp;

if(i==0)printf("COWCULATIONS OUTPUT\n");

str="";

num1=convertir(num1);

num2=convertir(num2);

resp=convertir(resp);

num2=opcion(s1,num1,num2);

num2=opcion(s2,num1,num2);

num2=opcion(s3,num1,num2);

num2=completar(str,num2);

if(num2.compare(resp)==0)printf("YES\n");

else printf("NO\n");

if(i==n-1) printf("END OF OUTPUT\n");

}

return 0;

}

# Intersecting Lines

\*\*\*ID: 378

\*\*\*Tipo: Geometry, Lines.

#include <bits/stdc++.h>

#include<stdio.h>

#include<math.h>

using namespace std;

#define EPS 1e-9

struct Point{

Point(double x1,double y1){

x=x1;

y=y1;

}

Point(){

}

double x;

double y;

///PRODUCTO PUNTO

double operator \*(const Point &p1)const{

return x\*p1.x+y\*p1.y;

}

///PRODUCTO CRUZ

double operator ^(const Point &p1)const{

return x\*p1.y-y\*p1.x;

}

Point operator-(const Point &p1)const{

Point a(x-p1.x,y-p1.y);

return a;

}

bool operator =(const Point &p1)const{

return (x==p1.x && y==p1.y );

}

};

void line\_line\_intersection(Point A,Point B,Point C,Point D){

Point CD=C-D, CA=A-C,AB=B-A,AC=A-C;

double t0=CD^CA;

double t1=AB^AC;

double det=AB^CD;

if (fabs(det) < EPS){

//paralelos

if (fabs(t0) < EPS || fabs(t1) < EPS){

//misma linea

printf("LINE\n");

}else{

//paralelas

printf("NONE\n");

}

}else{

t0 /= det;

t1 /= det;

double x = A.x + t0\*(B.x-A.x);

double y = A.y + t0\*(B.y-A.y);

//interseccion de puntos (x, y)

printf("POINT %.2lf %.2lf\n", x, y);

}

}

int main(){

int casos;

double x1,y1,x2,y2,x3,y3,x4,y4;

bool first=true;

scanf("%d",&casos);

printf("INTERSECTING LINES OUTPUT\n");

while(casos--){

scanf("%lf %lf %lf %lf %lf %lf %lf %lf",&x1,&y1,&x2,&y2,&x3,&y3,&x4,&y4);

Point p1(x1,y1),p2(x2,y2);

Point p3(x3,y3),p4(x4,y4);

line\_line\_intersection(p1,p2,p3,p4);

}

printf("END OF OUTPUT\n");

return 0;

}

# Perfection

\*\*\*ID: 382

\*\*\*Tipo: Ad hoc, number theory

#include<stdio.h>

int main(){

int n,i,aux,sum,c=1;

while(scanf("%d",&n) && n!=0){

if(c==1)printf("PERFECTION OUTPUT\n");

sum=0;

aux=n;

for(i=1;i<aux;i++){

if(n%i==0){

sum+=i;

}

}

if(sum==aux) printf("%5d PERFECT\n",aux);

else if(sum<aux)printf("%5d DEFICIENT\n",aux);

else if(sum>aux)printf("%5d ABUNDANT\n",aux);

c++;

}

printf("END OF OUTPUT\n");

return 0;

}

# Shipping Routes

\*\*\*ID: 383

\*\*\*Tipo: Graph Theory, BFS, Shortest path

#include <bits/stdc++.h>

#include<stdio.h>

#include<queue>

#include<map>

using namespace std;

#define MAX 40

int ady[MAX][MAX];

int visitado[MAX];

int V;

struct Estado{

int x;

int d;

Estado(int x1,int d1):x(x1),d(d1){}

};

int bfs(int ini,int fin){

Estado inicial(ini,0);

queue<Estado> Q;

Q.push(inicial);

memset(visitado,false,sizeof(visitado));

while(!Q.empty()){

Estado actual=Q.front();Q.pop();

if(actual.x==fin){return actual.d;}

visitado[actual.x]=true;

for(int i=0;i<V;i++){

if(ady[actual.x][i]==1 && !visitado[i]){

Estado vecino(i,actual.d+1);

Q.push(vecino);

}

}

}

return -1;

}

int main(){

int casos,A,R,costo;

string s,s2,s1;

scanf("%d",&casos);

printf("SHIPPING ROUTES OUTPUT\n");

for(int i=1;i<=casos;i++){

putchar('\n');

printf("DATA SET %d\n\n",i);

scanf("%d %d %d",&V,&A,&R);

map<string,int> mp;

for(int j=0;j<V;j++){

cin>>s;

mp[s]=j;

}

for(int i=0;i<=V;i++){

for(int j=0;j<=V;j++)ady[i][j]=0;

}

while(A--){

cin>>s1>>s2;

ady[mp[s1]][mp[s2]]=1;

ady[mp[s2]][mp[s1]]=1;

}

while(R--){

cin>>costo>>s1>>s2;

int time=bfs(mp[s1],mp[s2]);

if(time==-1){

printf("NO SHIPMENT POSSIBLE\n");

}

else printf("$%d\n",time\*costo\*100);

}

}

printf("\nEND OF OUTPUT\n");

return 0;

}

# Perfect Cubes

\*\*\*ID: 386

\*\*\*Tipo: Ad hoc, Simulation, no input

#include<stdio.h>

int main(){

for( int i = 6 ; i <= 200 ; ++i ){

for(int j = 2 ; j <= 150 ; ++j ){

for(int k = j+1 ; k <= 150 ; ++k ){

for(int q = k+1 ; q <= 200 ; ++q ){

long long sum = j \* j \* j + k \* k \* k + q \* q \* q;

if( sum == i \* i \* i ){

printf("Cube = %d, Triple = (%d,%d,%d)\n", i, j, k, q);

}

}

}

}

}

return 0;

}

# Basically Speaking

\*\*\*ID: 389

\*\*\*Tipo: Number Theory, Base conversion

#include <bits/stdc++.h>

#include<stdio.h>

#include<string>

#include<sstream>

using namespace std;

int valor(char n){

int j,i;

for(i='A',j=10;i<='F' && j<=15;i++,j++){

if(n==i)break;

}

return j;

}

char valor2(int n){

int i,j;

for(i=10,j='A';i<=15 && j<='F';i++,j++){

if(n==i)break;

}

return (char)j;

}

long long pow(long long a,long long b){

long long r=1;

for(int i=0;i<b;i++){

r\*=a;

}

return r;

}

long long base(string n,int b){

int i,idx=0,aux;

long long r=0;

for(i=n.length()-1;i>=0;i--){

if(isdigit(n[i])){

aux=n[i]-'0';

}

else{

aux=valor(n[i]);

}

if(aux>=b)return -1;

r+=(aux\*pow(b,idx));

idx++;

}

return r;

}

string toString(long long n){

string s;

ostringstream buffer;

buffer<<n;

s= buffer.str();

return s;

}

///DIVISIONES SUCESIVAS

string base2(long long n,long long b){

string str="";

int aux;

if(n==0)return "0";

while(n>=1){

aux=n%b;

if(aux>9 && aux<16){str=valor2(aux)+str;}

else {str=toString(aux)+str;}

n/=b;

}

return str;

}

int main(){

string n,resp,str;

long long bi,bf,d;

while(cin>>n>>bi>>bf){

d=base(n,bi);

str="";

if(d==-1)cout<<"ERROR"<<endl;

else{

resp=base2(d,bf);

if(resp.length()>7) resp="ERROR";

for(int i=0;i<7-resp.length();i++){

str+=" ";

}

str+=resp;

cout<<str<<endl;

}

}

return 0;

}

# Mapmaker

\*\*\*ID: 394

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

struct Estado{

int B;

int D;

int CD;

vector<int> U;

vector<int> L;

Estado(){};

};

int main(){

int n, m , B, D, L, U, resp,CD;

string s;

map< string , Estado > mp;

cin>>n>>m;

while( n-- ){

cin>>s>>B>>CD>>D;

Estado nuevo;

nuevo.B = B;

nuevo.D = D;

nuevo.CD = CD;

nuevo.U.push\_back(0);

nuevo.L.push\_back(0);

while( D-- ){

cin>>L>>U;

//vector de pares

nuevo.U.push\_back(U);

nuevo.L.push\_back(L);

}

mp[s]=nuevo;

}

while( m-- ){

cin>>s;

Estado actual = mp[s];

int i [ actual.B ], C[ actual.D + 1 ];

for(int j = 1 ; j <= actual.D ; ++j ){

cin>>i[j];

}

//SOLUCION

//Cd

C[ actual.D ] = actual.CD;

for( int j = actual.D - 1 ; j >= 1 ; --j ){

C[j] = C[j+1]\* ( actual.U[j+1] - actual.L[j+1] + 1);

}

C[0] = actual.B;

for ( int j = 1 ; j <= actual.D ; ++j ){

C[0]-=(C[j]\*actual.L[j]);

}

resp = C[0];

for( int j = 1;j <=actual.D ; ++j){

resp+= C[j]\*i[j];

}

//IMPRESION DE RESPUESTA

cout<<s<<"["<<i[1];

for(int j = 2 ; j <= actual.D; ++j ){

cout<<", "<<i[j];

}

cout<<"] = "<<resp<<endl;

}

return 0;

}.

# Equation Elation

\*\*\*ID: 397

\*\*\*Tipo: String Parsing

#include <bits/stdc++.h>

using namespace std;

#define MAX 1000

char s[ MAX ];

char token[ MAX ];

int idx , type , num , ant;

int ten[ ] = { 1 , 10 ,100 , 1000 , 10000 , 100000 , 1000000 , 10000000 };

bool isDigit( char c ){ return ( c >= '0' && c <= '9'); }

bool isLetter(char c){return c>='A' && c<='Z' || c>='a' && c<='z';}

/\*

Separo el string en tokens con sus tipos

\*/

void next(){

int len = 0 , i , j;

//delete spaces

token[ 0 ] = '\0';

for( ; s[ idx ] ; ++idx ){ if( s[ idx ] != ' ' ) break; }

if( isLetter( s[ idx ] ) ){

for( ; s[ idx ] ; ++idx ){

if( !isLetter( s[ idx ] ) )break;

token[ len++ ] = s[ idx ];

}

token[ len ] = '\0';

return;

}

if( !isDigit( s[ idx ] ) ){

token[ len++ ] = s[ idx++ ];

if( !isDigit( s[ idx ] ) || ( isDigit( s[ idx ] ) && ant == 'n' ) ){ type = s[ idx - 1 ]; return; }

}

for( ; s[ idx ] ; ++idx ){

if( isDigit( s[ idx ] ) ) token[ len++ ] = s[ idx ];

else break;

}

if( len != 0 ){

int ini = ( isDigit( token[ 0 ] ) )?0:1;

num = 0;

for( i = len - 1 , j = 0 ; i >= ini ; --i ){

num = ( token[ i ] - '0' ) \* ten[ j++ ] + num;

}

type = 'n';

if( ini ){

if( token[ 0 ] == '-' )num = -num;

}

return;

}

}

#define pb push\_back

vector< int > v;

vector< int > types;

char end[ MAX ];

/\*

Guardo en vector v los valores y en vector types los tipos de cada valor si es entero o signo

\*/

void parse(){

types.clear();

v.clear();

next();

ant = type;

v.pb( num );

types.pb( type );

while( type != '=' ){

next();

ant = type;

if( type == '=' ) break;

v.pb( type );

types.pb( type );

next();

ant = type;

v.pb( num );

types.pb( type );

}

v.pb( type );

types.pb( type );

next();

memcpy( end , token , sizeof( token ) );

}

void print(){

int i , n = v.size() ;

printf("%d" , v[ 0 ] );

for( i = 1 ; i < n ; ++i ){

if( i & 1 ) printf(" %c" , v[ i ] );

else printf(" %d" , v[ i ] );

}

printf(" %s\n" , end );

}

/\*

Busco iterando algun \* o / y opero si no tengo esos signos

busco + o - y opera sino termino.

\*/

void solve(){

int i , n = v.size() , j , signo;

vector< int > aux , auxT;

while( true ){

print();

n = v.size();

aux.clear();

auxT.clear();

signo = -1;

for( i = 0 ; i < n ; ++i ){

if( types[ i ] == '\*' || types[ i ] == '/' ){

signo = types[ i ];

i--;

break;

}

}

if( signo == '\*' || signo == '/' ){

for( j = 0 ; j < i ; ++j ){

aux.pb( v[ j ] );

auxT.pb( types[ j ] );

}

if( signo == '\*' ){

aux.pb( v[ i ] \* v[ i + 2 ] );

auxT.pb( 'n' );

}

else{

aux.pb( v[ i ]/v[ i + 2 ]);

auxT.pb( 'n' );

}

for( j = i + 3 ; j < n ; ++j ){

aux.pb( v[ j ] );

auxT.pb( types[ j ] );

}

}

else{

for( i = 0 ; i < n ; ++i ){

if( types[ i ] == '+' || types[ i ] == '-' ){

signo = types[ i ];

i--;

break;

}

}

if( signo == '+' || signo == '-' ){

for( j = 0 ; j < i ; ++j ){

aux.pb( v[ j ] );

auxT.pb( types[ j ] );

}

if( signo == '+' ){

aux.pb( v[ i ] + v[ i + 2 ] );

auxT.pb( 'n' );

}

else{

aux.pb( v[ i ] - v[ i + 2 ]);

auxT.pb( 'n' );

}

for( j = i + 3 ; j < n ; ++j ){

aux.pb( v[ j ] );

auxT.pb( types[ j ] );

}

}

else break;

}

v = aux;

types = auxT;

}

}

int main(){

bool b = false;

while( gets( s ) ){

if( b ) printf("\n");

b = true;

idx = 0;

ant = -1;

parse();

solve();

}

return 0;

}

# Palindromes

\*\*\*ID: 401

\*\*\*Tipo: Ad hoc, strings, palindromes

#include <bits/stdc++.h>

#include<stdio.h>

#include<cstring>

using namespace std;

bool isPalindrome(char \*c,int l){

for(int i=0,j=l-1;i<=l/2 && j>=l/2;i++,j--){

if(c[i]!=c[j])return false;

}

return true;

}

bool isMirror(char \*c,int l){

for(int i=0,j=l-1;i<=l/2 && j>=l/2;i++,j--){

if(c[i]=='A' && c[j]=='A')continue;

if(c[i]=='E' && c[j]=='3')continue;

if(c[i]=='3' && c[j]=='E')continue;

if(c[i]=='H' && c[j]=='H')continue;

if(c[i]=='I' && c[j]=='I')continue;

if(c[i]=='J' && c[j]=='L')continue;

if(c[i]=='L' && c[j]=='J')continue;

if(c[i]=='M' && c[j]=='M')continue;

if(c[i]=='O' && c[j]=='O')continue;

if(c[i]=='S' && c[j]=='2')continue;

if(c[i]=='T' && c[j]=='T')continue;

if(c[i]=='U' && c[j]=='U')continue;

if(c[i]=='V' && c[j]=='V')continue;

if(c[i]=='W' && c[j]=='W')continue;

if(c[i]=='X' && c[j]=='X')continue;

if(c[i]=='Y' && c[j]=='Y')continue;

if(c[i]=='Z' && c[j]=='5')continue;

if(c[i]=='5' && c[j]=='Z')continue;

if(c[i]=='1' && c[j]=='1')continue;

if(c[i]=='2' && c[j]=='S')continue;

if(c[i]=='8' && c[j]=='8')continue;

else return false;

}

return true;

}

int main(){

char s[25];

int l;

bool ip,im;

while(gets(s)){

l=strlen(s);

ip=isPalindrome(s,l);

im=isMirror(s,l);

if(!ip && !im) printf("%s -- is not a palindrome.\n\n",s);

else if(ip && !im)printf("%s -- is a regular palindrome.\n\n",s);

else if(!ip && im)printf("%s -- is a mirrored string.\n\n",s);

else if(ip && im) printf("%s -- is a mirrored palindrome.\n\n",s);

}

return 0;

}

# M\*A\*S\*H

\*\*\*ID: 402

\*\*\*Tipo: Simulation

#include <bits/stdc++.h>

#include <vector>

using namespace std;

int main(){

int n , l , deck[ 21 ] , q = 1;

bool first;

while( scanf("%d %d" , &n , &l ) == 2 ){

vector<int> Circle;

for( int i = 0 ; i < 20 ; ++i )scanf("%d" , &deck[ i ] );

for( int i = 1 ; i <= n ; ++i ) Circle.push\_back( i );

for( int card = 0, persons = n ; persons > l && card < 20 ; ++card ){

//para cada carta hago josehpus hasta la persona N

int i = 0, next;

while( i < Circle.size() && persons > l ){

next = ( i + deck[ card ] - 1 );

if( next >= Circle.size() || persons <= l ) break;

Circle.erase( Circle.begin() + next );

persons--;

i = next;

}

}

first = true;

printf("Selection #%d\n" , q++ );

for( int i = 0 ; i < Circle.size() ; ++i ){

if( !first ) printf(" ");

printf("%d" , Circle[ i ] );

first = false;

}

printf("\n\n");

}

return 0;

}

# Prime Cuts

\*\*\*ID: 406

\*\*\*Tipo: Number Theory, Sieve, Primes

#include <bits/stdc++.h>

using namespace std;

#define MAX 1001

int prime[MAX];

void sieve(){

int i,k;

for(i = 1; i <= MAX; ++i){

prime[i]=1;

}

for(i=2;i\*i<=MAX;i++){

if(prime[i]==1){

for(k=2;k\*i<=MAX;k++){

prime[k\*i]=0;

}

}

}

}

int main(){

sieve();

int a,c,i,id,cant;

while(scanf("%d %d",&a,&c)==2){

int primes[a];

id=0;

printf("%d %d:",a,c);

for(i=1;i<=a;i++){if(prime[i]==1){primes[id++]=i;}}

(id%2==0)?c\*=2: c=c\*2-1;

if(c>id)c=id;

cant=(id-c)/2;

if(cant<0)cant=0;

for(int i=cant;i<cant+c;++i)

printf(" %d", primes[i]);

printf("\n\n");

}

return 0;

}

# Uniform Generator

\*\*\*ID: 408

\*\*\*Tipo: Cycle finding

#include <bits/stdc++.h>

#include<cstdio>

using namespace std;

int main()

{

int s, m, now, i;

while( scanf("%d %d", &s ,&m ) == 2 ) {

int a[ 100005 ] = { 0 };

now = s % m; a[ now ] = 1;

while( now ) {

now = ( now + s ) % m;

a[ now ] = 1;

}

for( i = 0 ; i < m ; i++)

if( !a[ i ] ) break;

if( i >= m ) printf("%10d%10d Good Choice\n\n", s, m);

else printf("%10d%10d Bad Choice\n\n", s, m);

}

return 0;

}

# Station Balance

\*\*\*ID: 410

\*\*\*Tipo: Greedy

#include <bits/stdc++.h>

using namespace std;

#define MAX 20

int a[ MAX ];

int main(){

int C, N, alone , q = 1, T , ini;

double IMB, prom;

while( scanf("%d %d" , &C , &N ) == 2 ){

memset( a , 0 , sizeof( a ) );

IMB = 0.0;

prom = 0.0;

ini = (C<<1) - N;

for( int i = ini ; i < (C<<1) ; ++i ){

scanf("%d" , &a[ i ] );

prom += a[ i ];

}

sort( a , a + ( C<<1 ) );

printf("Set #%d\n" , q++ );

for( int i = 0 ; i < C; ++i ){

printf(" %d:" , i );

IMB += fabs( C \* a[ i ] + C \* a[ ( C<<1 ) - i - 1 ] - prom );

if( a[ i ] ){

printf(" %d" , a[ i ] );

}

if( a[ ( C<<1 ) - i - 1 ] )printf(" %d" , a[ ( C<<1 ) - i - 1 ] );

printf("\n");

}

printf("IMBALANCE = %.5lf\n\n" , IMB/(double)C );

}

return 0;

}

# Pi

\*\*\*ID: 412

\*\*\*Tipo: Ad hoc, simulation, number theory, gcd

#include <bits/stdc++.h>

#include<stdio.h>

#include<math.h>

using namespace std;

int gcd(int a,int b){

return (b==0)? a:gcd(b,a%b);

}

int main(){

int n,i,c,j,cont,acu;

while(scanf("%d",&n)!=EOF){

if(n==0)break;

cont=0;

acu=0;

int casos[n];

for(i=0;i<n;i++){

scanf("%d",&c);

casos[i]=c;

}

for(i=0;i<n;i++){

for(j=i+1;j<n;j++){

if(gcd(casos[i],casos[j])==1)cont++;

acu++;

}

}

if(cont==0) printf("No estimate for this data set.\n");

else{ printf("%0.6lf\n",sqrt(6.0\*acu/cont));

}

}

return 0;

}

# Machined Surfaces

\*\*\*ID: 414

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

int main(){

int n,maxi,resp;

string s;

while(scanf("%d",&n) && n!=0){

int a[n];

memset(a,0,sizeof(a));

maxi=0;

resp=0;

cin.get();

for(int j=0;j<n;j++){

getline(cin,s);

for(int i=0;i<s.length();i++){

if(s[i]=='X'){a[j]++;}

}

maxi=max(maxi,a[j]);

}

for(int i=0;i<n;i++){

resp+=maxi-a[i];

}

printf("%d\n",resp);

}

return 0;

}

# LED Test

\*\*\*ID: 416

\*\*\*Tipo: Backtracking

#include <bits/stdc++.h>

using namespace std;

#define MAX 12

char led[ MAX ][ 9 ] = { "YYYYYYN", "NYYNNNN", "YYNYYNY", "YYYYNNY",

"NYYNNYY", "YNYYNYY", "YNYYYYY", "YYYNNNN",

"YYYYYYY", "YYYYNYY" };

int n;

string s[ MAX ];

bool possible[ 15 ][ 15 ];

bool match;

void solve( int len , int value , int badMask ){

if( len == n ){

match = true;

return;

}

if( possible[ len ][ value ] ){

for( int i = 0 ; i < 7 ; ++i ){

if( s[len][i] == 'Y' && (badMask & ( 1<<i) ) )

return;

}

int mask = badMask;

for( int i = 0 ; i < 7 ; ++i ){

if( s[ len ][ i ] == 'N' && led[ value ][ i ] == 'Y' ){

mask |= 1<<i;

}

}

solve( len + 1 , value - 1 , mask );

}

}

int main(){

while( scanf("%d" , &n ) , n ){

memset( possible , 0 , sizeof( possible ) );

for( int i = 0 ; i < n && cin>>s[i]; ++i ){

for( int j = 0 ; j < 10 ; ++j ){

int k = 0;

for( ; k < 7 ; ++k ){

if( s[ i ][ k ] == 'Y' && led[ j ][ k ] == 'N')break;

}

if( k == 7 ) {

possible[i][j] = true;

}

}

}

match = false;

for( int j = 9 ; j >= n - 1 && !match; --j ){

solve( 0 , j , 0 );

}

if( match )

puts("MATCH");

else

puts("MISMATCH");

}

return 0;

}

# Word Index

\*\*\*ID: 417

\*\*\*Tipo: Ad hoc, brute force

#include <bits/stdc++.h>

#include<stdio.h>

#include<map>

using namespace std;

map<string,int> mp;

void solve(){

string s;

int num=1;

for(int i='a';i<='z';i++){

s="";

s+=(char)i;

mp[s]=num++;

}

for(int i='a';i<='z';i++){

for(int j=i+1;j<='z';j++){

s="";

s+=(char)i;

s+=(char)j;

mp[s]=num++;

}

}

for(int i='a';i<='z';i++){

for(int j=i+1;j<='z';j++){

for(int k=j+1;k<='z';k++){

s="";

s+=(char)i;

s+=(char)j;

s+=(char)k;

mp[s]=num++;

}

}

}

for(int i='a';i<='z';i++){

for(int j=i+1;j<='z';j++){

for(int k=j+1;k<='z';k++){

for(int q=k+1;q<='z';q++){

s="";

s+=(char)i;

s+=(char)j;

s+=(char)k;

s+=(char)q;

mp[s]=num++;

}

}

}

}

for(int i='a';i<='v';i++){

for(int j=i+1;j<='z';j++){

for(int k=j+1;k<='z';k++){

for(int q=k+1;q<='z';q++){

for(int w=q+1;w<='z';w++){

s="";

s+=(char)i;

s+=(char)j;

s+=(char)k;

s+=(char)q;

s+=(char)w;

mp[s]=num++;

}

}

}

}

}

}

int main(){

solve();

string s;

while(cin>>s){

cout<<mp[s]<<endl;

}

return 0;

}

# Word-Search Wonder

\*\*\*ID: 422

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

#define MAX 105

char ady[ MAX ][ MAX ] , s[ MAX ];

int n , len;

int dx[ 7 ] = { 1 , 0 , 0 , 1 , 1 , -1 , -1 };

int dy[ 7 ] = { 0 , 1 , -1 , 1 , -1 , 1 , -1};

bool solve( int x , int y){

int i , k , nx , ny;

for( i = 0 ; i < 7 ; ++i ){

k = 0;

while( true ){

nx = dx[ i ] \* k + x; ny = dy[ i ] \* k + y;

if( nx >= 0 && ny >= 0 && nx < n && ny < n && ady[ nx ][ ny ] == s[ k ] ){

k++;

}

else break;

if( k == len ){

printf("%d,%d %d,%d\n" , x + 1 , y + 1 , nx + 1 , ny + 1 );

return true;

}

}

}

return false;

}

int main(){

int i, j;

scanf("%d" , &n );

for( i = 0 ; i < n && scanf("%s" , &ady[ i ] ) ; ++i );

while( scanf("%s" , &s) , s[ 0 ] != '0' ){

len = strlen( s );

for( i = 0 ; i < n ; ++i ){

for( j = 0 ; j < n ; ++j ){

if( solve( i , j ) ){ goto end;}

}

}

end:

if( i == n ) puts("Not found");

}

return 0;

}

# MPI Maelstrom

\*\*\*ID: 423

\*\*\*Tipo: Graph Theory, Floyd Warshall

#include <bits/stdc++.h>

using namespace std;

#define MAX 105

#define INF 9999999

int ady[MAX][MAX] , V;

void Init(){

for(int i = 0 ; i < MAX ; ++i ){

for(int j = 0 ; j < MAX ; ++j ){

ady[ i ][ j ] = INF;

}

ady[ i ][ i ] = 0;

}

}

void floyd(){

for(int k = 0 ; k < V ; ++k ){

for(int i = 0 ; i < V ; ++i ){

for(int j = 0 ; j < V ; ++j ){

int t = ady[ i ][ k ] + ady[ k ][ j ];

if( t < ady[ i ][ j ] ){

ady[ i ][ j ] = t;

}

}

}

}

}

int toInt( string s ){

stringstream ss( s);

int n;

ss>>n;

return n;

}

int main(){

int j;

string s;

scanf("%d" , &V);

Init();

cin.get();

for( int i = 1 ; i < V ;++i){

getline( cin , s );

s+=" ";

stringstream ss( s );

j = 0;

while( ss>> s ){

if( s != "x" ){

ady[ i ][ j ] = ady[ j ][ i ] = toInt( s );

}

j++;

}

}

floyd();

int resp = 0;

for( int i = 0 ; i < V ; ++i ){

if( ady[ 0 ][ i ] != INF && resp < ady[ 0 ][ i ])resp = ady[ 0 ][ i ];

}

printf("%d\n" , resp);

return 0;

}

# Integer Inquiry

\*\*\*ID: 424

\*\*\*Tipo: Bignum, Addition

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

const int MAXD = 105, DIG = 9, BASE = 1000000000;

const unsigned long long BOUND = numeric\_limits <unsigned long long> :: max () - (unsigned long long) BASE \* BASE;

struct bignum

{

int D, digits [MAXD / DIG + 2];

inline void trim ()

{

while (D > 1 && digits [D - 1] == 0)

D--;

}

inline void init (long long x)

{

memset (digits, 0, sizeof (digits));

D = 0;

do

{

digits [D++] = x % BASE;

x /= BASE;

}

while (x > 0);

}

inline bignum (long long x)

{

init (x);

}

inline bignum (int x = 0)

{

init (x);

}

inline bignum (char \*s)

{

memset (digits, 0, sizeof (digits));

int len = strlen (s), first = (len + DIG - 1) % DIG + 1;

D = (len + DIG - 1) / DIG;

for (int i = 0; i < first; i++)

digits [D - 1] = digits [D - 1] \* 10 + s [i] - '0';

for (int i = first, d = D - 2; i < len; i += DIG, d--)

for (int j = i; j < i + DIG; j++)

digits [d] = digits [d] \* 10 + s [j] - '0';

trim ();

}

inline char \*str ()

{

trim ();

char \*buf = new char [DIG \* D + 1];

int pos = 0, d = digits [D - 1];

do

{

buf [pos++] = d % 10 + '0';

d /= 10;

}

while (d > 0);

reverse (buf, buf + pos);

for (int i = D - 2; i >= 0; i--, pos += DIG)

for (int j = DIG - 1, t = digits [i]; j >= 0; j--)

{

buf [pos + j] = t % 10 + '0';

t /= 10;

}

buf [pos] = '\0';

return buf;

}

inline bool operator < (const bignum &o) const

{

if (D != o.D)

return D < o.D;

for (int i = D - 1; i >= 0; i--)

if (digits [i] != o.digits [i])

return digits [i] < o.digits [i];

return false;

}

inline bool operator == (const bignum &o) const

{

if (D != o.D)

return false;

for (int i = 0; i < D; i++)

if (digits [i] != o.digits [i])

return false;

return true;

}

inline bignum operator << (int p) const

{

bignum temp;

temp.D = D + p;

for (int i = 0; i < D; i++)

temp.digits [i + p] = digits [i];

for (int i = 0; i < p; i++)

temp.digits [i] = 0;

return temp;

}

inline bignum operator >> (int p) const

{

bignum temp;

temp.D = D - p;

for (int i = 0; i < D - p; i++)

temp.digits [i] = digits [i + p];

for (int i = D - p; i < D; i++)

temp.digits [i] = 0;

return temp;

}

inline bignum range (int a, int b) const

{

bignum temp = 0;

temp.D = b - a;

for (int i = 0; i < temp.D; i++)

temp.digits [i] = digits [i + a];

return temp;

}

inline bignum operator + (const bignum &o) const

{

bignum sum = o;

int carry = 0;

for (sum.D = 0; sum.D < D || carry > 0; sum.D++)

{

sum.digits [sum.D] += (sum.D < D ? digits [sum.D] : 0) + carry;

if (sum.digits [sum.D] >= BASE)

{

sum.digits [sum.D] -= BASE;

carry = 1;

}

else

carry = 0;

}

sum.D = max (sum.D, o.D);

sum.trim ();

return sum;

}

inline bignum operator - (const bignum &o) const

{

bignum diff = \*this;

for (int i = 0, carry = 0; i < o.D || carry > 0; i++)

{

diff.digits [i] -= (i < o.D ? o.digits [i] : 0) + carry;

if (diff.digits [i] < 0)

{

diff.digits [i] += BASE;

carry = 1;

}

else

carry = 0;

}

diff.trim ();

return diff;

}

inline bignum operator \* (const bignum &o) const

{

bignum prod = 0;

unsigned long long sum = 0, carry = 0;

for (prod.D = 0; prod.D < D + o.D - 1 || carry > 0; prod.D++)

{

sum = carry % BASE;

carry /= BASE;

for (int j = max (prod.D - o.D + 1, 0); j <= min (D - 1, prod.D); j++)

{

sum += (unsigned long long) digits [j] \* o.digits [prod.D - j];

if (sum >= BOUND)

{

carry += sum / BASE;

sum %= BASE;

}

}

carry += sum / BASE;

prod.digits [prod.D] = sum % BASE;

}

prod.trim ();

return prod;

}

inline double double\_div (const bignum &o) const

{

double val = 0, oval = 0;

int num = 0, onum = 0;

for (int i = D - 1; i >= max (D - 3, 0); i--, num++)

val = val \* BASE + digits [i];

for (int i = o.D - 1; i >= max (o.D - 3, 0); i--, onum++)

oval = oval \* BASE + o.digits [i];

return val / oval \* (D - num > o.D - onum ? BASE : 1);

}

inline pair <bignum, bignum> divmod (const bignum &o) const

{

bignum quot = 0, rem = \*this, temp;

for (int i = D - o.D; i >= 0; i--)

{

temp = rem.range (i, rem.D);

int div = (int) temp.double\_div (o);

bignum mult = o \* div;

while (div > 0 && temp < mult)

{

mult = mult - o;

div--;

}

while (div + 1 < BASE && !(temp < mult + o))

{

mult = mult + o;

div++;

}

rem = rem - (o \* div << i);

if (div > 0)

{

quot.digits [i] = div;

quot.D = max (quot.D, i + 1);

}

}

quot.trim ();

rem.trim ();

return make\_pair (quot, rem);

}

inline bignum operator / (const bignum &o) const

{

return divmod (o).first;

}

inline bignum operator % (const bignum &o) const

{

return divmod (o).second;

}

inline bignum power (int exp) const

{

bignum p = 1, temp = \*this;

while (exp > 0)

{

if (exp & 1) p = p \* temp;

if (exp > 1) temp = temp \* temp;

exp >>= 1;

}

return p;

}

};

inline bignum gcd (bignum a, bignum b)

{

bignum t;

while (!(b == 0))

{

t = a % b;

a = b;

b = t;

}

return a;

}

int main(){

char s[105];

bignum sum=0;

while(cin>>s && strcmp(s,"0")!=0){

bignum n(s);

sum=sum+n;

}

cout<<sum.str()<<endl;

return 0;

}

# Word Transformation

\*\*\*ID: 429

\*\*\*Tipo: Graph Theory, BFS, strings

#include <bits/stdc++.h>

using namespace std;

#define MAX 201

#define INF 9999999

int ady[MAX][MAX];

int V;

void init(){

for(int i=0;i<=V;i++){

for(int j=0;j<=V;j++){

ady[i][j]=INF;

}

ady[i][i]=0;

}

}

void floyd(){

for(int k=0;k<V;k++){

for(int i=0;i<V;i++){

for(int j=0;j<V;j++){

int t=ady[i][k]+ady[k][j];

if(t<ady[i][j]){

ady[i][j]=t;

}

}

}

}

}

int main(){

int casos,index;

scanf("%d",&casos);

string s,s2,s1;

char C;

for(int x=0;x<casos;x++){

if(x!=0){putchar('\n');}

vector<string> dictionary;

map<string,int> mp;

index=0;

while(cin>>s && s!="\*"){

dictionary.push\_back(s);

mp[s]=index++;

}

V=index;

init();

///Formo grafo conenctando palabras con un caracter de diferencia

int n=dictionary.size();

for(int i=0;i<n;i++){

s=dictionary[i];

for(int k=i+1;k<n;k++){

s2=dictionary[k];

if(s2.length()!=s.length())continue;

int diferencia=0;

for(int q=0;q<s2.length();q++){

if(s[q]!=s2[q])diferencia++;

if(diferencia>1)break;

}

//si se diferencia en un caracter hago el enlace

if(diferencia==1){

ady[mp[s]][mp[s2]]=1;

ady[mp[s2]][mp[s]]=1;

}

}

}

floyd();

cin.get();

while(getline(cin,s1)){

if(s1.length()==0)break;

stringstream ss(s1);

ss>>s>>s2;

if(ady[mp[s]][mp[s2]]==INF)cout<<s<<" "<<s2<<" 0"<<endl;

else cout<<s<<" "<<s2<<" "<<ady[mp[s]][mp[s2]]<<endl;

}

}

return 0;

}

# Bank (Not Quite O.C.R.)

\*\*\*ID: 433

\*\*\*Tipo: Ad hoc, Brute Force

#include <bits/stdc++.h>

using namespace std;

char led[ 10 ][ 9 ] = { "YYYYYYN", "NYYNNNN", "YYNYYNY", "YYYYNNY",

"NYYNNYY", "YNYYNYY", "YNYYYYY", "YYYNNNN",

"YYYYYYY", "YYYYNYY" };

char s1[ 305 ], s2[ 305 ] , s3[ 305 ];

bool possible[ 15 ][ 15 ];

int dif[ 15 ][ 15 ], real[ 15 ];

void toInt(){

int l = strlen( s1 );

for( int i = l ; i <= 27 ; ++i ) s1[ i ] = ' ';

l = strlen( s2 );

for( int j = l ; j <= 27 ; ++j ) s2[ j ] = ' ';

l = strlen( s3 );

for( int i = l ; i <= 27 ; ++i ) s3[ i ] = ' ';

memset( possible , 0 , sizeof( possible ) );

memset( real , -1 , sizeof( real ) );

memset( dif , 0 , sizeof( dif ) );

for( int i = 0 , k = 0 ; i < 27 ; i += 3 ){

for( int j = 0 ; j < 10 ; ++j ){

if( ( led[ j ][ 0 ] == 'N' && ( s1[ i + 1 ] == '\_' ) ) ||

( led[ j ][ 1 ] == 'N' && ( s2[ i + 2 ] == '|' ) ) ||

( led[ j ][ 2 ] == 'N' && ( s3[ i + 2 ] == '|' ) ) ||

( led[ j ][ 3 ] == 'N' && ( s3[ i + 1 ] == '\_' ) ) ||

( led[ j ][ 4 ] == 'N' && ( s3[ i ] == '|' ) ) ||

( led[ j ][ 5 ] == 'N' && ( s2[ i ] == '|' ) ) ||

( led[ j ][ 6 ] == 'N' && ( s2[ i + 1 ] == '\_') ) ){

possible[ k ][ j ] = 0;

}

else {

if( led[ j ][ 0 ] == 'Y' && s1[ i + 1 ] != '\_' ) { dif[ k ][ j ]++;}

if( led[ j ][ 1 ] == 'Y' && s2[ i + 2 ] != '|' ) dif[ k ][ j ]++;

if( led[ j ][ 2 ] == 'Y' && s3[ i + 2 ] != '|' ) dif[ k ][ j ]++;

if( led[ j ][ 3 ] == 'Y' && s3[ i + 1 ] != '\_' ) dif[ k ][ j ]++;

if( led[ j ][ 4 ] == 'Y' && s3[ i ] != '|' ) dif[ k ][ j ]++;

if( led[ j ][ 5 ] == 'Y' && s2[ i ] != '|' ) dif[ k ][ j ]++;

if( led[ j ][ 6 ] == 'Y' && s2[ i + 1 ] != '\_' ) dif[ k ][ j ]++;

if( dif[ k ][ j ] == 0 ) real[ k ] = j;

possible[ k ][ j ] = 1;

}

}

k++;

}

}

int a[ 10 ], len;

bool check(){

long long ans = 0 ;

for( int i = 8 , j = 1 ; i >= 0 ; --i , ++j ) ans += a[ i ] \* j;

if( ans % 11 == 0 ) return true;

return false;

}

int main(){

int t ,ans , cnt, resp[ 10 ];

scanf("%d" , &t );

getchar();

while( t-- ){

gets( s1 ); gets( s2 ); gets( s3 );

toInt();

bool b = true;

//caso 1: si el inicial no posee malos, entonces puedo revisar de golpe y dar respuesta

for( int i = 0 ; i < 9 ; ++i ){

if( real[ i ] == -1 ){

b= false;

break;

}

a[ i ] = real[ i ];

}

if( b && check() ){

for( int i = 0 ; i < 9 ; ++i ) printf("%d" , a[ i ] );

printf("\n");

continue;

}

cnt = 0;

ans = -1;

for( int i = 0 ; i < 9 ; ++i ){

len = 0;

b = true;

for( int j = 0 ; j < i ; ++j ) {

if( real[ j ] == -1 ){ b = false; break;}

a[ len++ ] = real[ j ];

}

if( !b )break;

//caso 2: solo puedo tener uno malo de todos

for( int j = 0 ; j < 10 ; ++j ){

if( possible[ i ][ j ] ){

a[ len ] = j;

for( int k = i + 1 , q = len + 1 ; k < 9 ; ++k , q++ ){

if( real[ k ] == -1 ){ b = false; break;}

a[ q ] = real[ k ];

}

if( !b ) break;

else if( check() ){

cnt++;

if( cnt > 1 ){

i = 9;

break;

}

for( int q = 0 ; q < 9 ; ++q ) resp[ q ] = a[ q ];

break;

}

}

}

}

if( cnt > 1 ) puts("ambiguous");

else if( cnt == 1 ){

for( int i = 0 ; i < 9 ; ++i ) printf("%d" , resp[ i ] );

printf("\n");

}

else puts("failure");

}

return 0;

}

# Block Voting

\*\*\*ID: 435

\*\*\*Tipo: Brute force

#include <bits/stdc++.h>

using namespace std;

int main(){

int t , n, a[ 25 ], mid , sum, ans[ 25 ];

scanf("%d" , &t );

while( t-- ){

scanf("%d" , &n );

sum = 0;

for( int i = 0 ; i < n ; ++i ){ scanf("%d" , &a[ i ] ); sum += a[ i ]; }

mid = sum/2;

memset( ans , 0 , sizeof( ans ) );

for( int i = 1 ; i < 1<<n ; ++i ){

sum = 0;

for( int j = 0 ; j < n ; ++j ){

if( ( 1<<j ) & i ){

sum += a[ j ];

}

}

if( sum > mid ){

for( int j = 0 ; j < n ; ++j ){

if( ( 1<<j ) & i ){

if( sum - a[ j ] <= mid ) ans[ j ]++;

}

}

}

}

for( int i = 0 ; i < n ; ++i ){

printf("party %d has power index %d\n" , i + 1 , ans[ i ] );

}

printf("\n");

}

return 0;

}

# Arbirage II

\*\*\*ID: 436

\*\*\*Tipo: Floyd Warshall

#include <bits/stdc++.h>

#include <string.h>

#define MAX 35

#define EPS 1e-7

int V;

double ady[ MAX ][ MAX ];

char s[ 105 ];

void init(){

int i , j;

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

ady[ i ][ j ] = ( i == j )? 1.0 : -1.0;

}

}

}

void floyd(){

int i , k , j;

for( k = 0 ; k < V ; ++k ){

for( i = 0 ; i < V ; ++i ){

if( ady[ i ][ k ] == -1.0 ) continue;

for( j = 0 ; j < V ; ++j ){

if( ady[ k ][ j ] == -1.0 ) continue;

if( ady[ i ][ k ] \* ady[ k ][ j ] > ady[ i ][ j ] ) ady[ i ][ j ] = ady[ i ][ k ] \* ady[ k ][ j ];

}

}

}

}

char names[ MAX ][ 105 ];

int num;

int getId(){

int i;

for( i = 0 ; i < num ; ++i ){

if( strcmp( names[ i ] , s ) == 0 ) return i;

}

strcpy( names[ num ] , s );

return num++;

}

int main(){

int E , u , v , q = 1 , i;

double w;

while( scanf("%d", &V ) , V ){

num = 0;

for( i = 0 ; i < V && scanf("%s" , s ) == 1 ; ++i );

init();

scanf("%d" , &E );

while( E-- ){

scanf("%s" , s );

u = getId();

scanf("%lf" , &w );

scanf("%s" , s );

v = getId();

ady[ u ][ v ] = w;

}

floyd();

for( i = 0 ; i < V ; ++i )

if( ady[ i ][ i ] > 1.0 + EPS ) break;

printf("Case %d: " , q++ );

if( i == V ) puts("No");

else puts("Yes");

}

return 0;

}

# The Tower of Babylon

\*\*\*ID: 437

\*\*\*Tipo: LIS, DP

#include <cstdio>

#include <vector>

#include <iostream>

using namespace std;

#define MAX 200

int n;

vector< vector<int> > v;

bool inside( vector<int> a , vector<int> b ){

return a[ 0 ] < b[ 0 ] && a[ 1 ] < b[ 1 ];

}

int val[ MAX ];

int main(){

int x , y ,z , ans , q = 1;

while( scanf("%d" , &n ) , n ){

v.clear();

for( int i = 0 ; i < n ; ++i ){

vector<int> vv( 3 );

scanf("%d %d %d" , &vv[ 0 ] , &vv[ 1 ] , &vv[ 2 ] );

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

do{

v.push\_back( vv );

}while( next\_permutation( vv.begin() , vv.end() ) );

}

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

n = v.size();

ans = 0;

for( int i = 0 ; i < n ; ++i ) val[ i ] = v[ i ][ 2 ];

for( int i = 0 ; i < n ; ++i ){

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

if( inside( v[ i ] , v[ j ] ) && val[ j ] < val[ i ] + v[ j ][ 2 ] ){

val[ j ] = val[ i ] + v[ j ][ 2 ];

}

}

}

for( int i = 0 ; i < n ; ++i )

ans = max( ans , val[ i ] );

printf("Case %d: maximum height = %d\n" , q++ , ans );

}

return 0;

}

# The Circumference of the Circle

\*\*\*ID: 438

\*\*\*Tipo: Circle throught three points

#include <bits/stdc++.h>

using namespace std;

#define EPS 1e-9

const double PI = 3.141592653589793;

struct Point{

double x , y;

Point( double X , double Y ):x(X) , y(Y){}

bool operator <( Point other ) const{

if( fabs( x - other.x ) < EPS ){

return y + EPS < other.y;

}

return x + EPS < other.x;

}

bool operator ==( Point other )const{

return ( fabs( x - other.x ) < EPS && fabs( y - other.y ) < EPS );

}

Point operator -(Point other)const{

return Point( x - other.x , y - other.y );

}

Point operator + (Point other)const{

return Point( x + other.x , y + other.y );

}

Point(){}

};

//Distance between two points

double dist( Point p1 , Point p2 ){

return hypot( p1.x - p2.x , p1.y - p2.y );

}

//////////////////////////////////////////////////////////////////////////////////////////////////////

struct Line{

double a , b , c;

Line( double A , double B , double C ): a(A),b(B),c(C){}

Line(){}

};

//Ax + By = C

bool pointToLine( Point p1 , Point p2 , Line &l ){

l.a = (p2.y - p1.y);

l.b = (p1.x - p2.x);

l.c = l.a \* p1.x + l.b \* p1.y;

return true;

}

void perpendicularLine( Line &l , Point mid ){

swap( l.a , l.b );

l.a = -l.a;

l.c = l.a \* mid.x + l.b \* mid.y;

}

//Returns the point of intersection of two lines

bool lineIntersection( Line l1 , Line l2 , Point &p ){

double det = l1.a \* l2.b - l2.a \* l1.b;

if( fabs( det ) < EPS ) return false;

p.x = (l1.c \* l2.b - l2.c \* l1.b)/det;

p.y = (l1.a \* l2.c - l2.a \* l1.c)/det;

if( fabs(p.x) < EPS ) p.x = 0.0;

if( fabs(p.y) < EPS ) p.y = 0.0;

return true;

}

bool circle3Point( Point p1 , Point p2 , Point p3 , Point &c ){

Point mid( ( p1.x + p2.x)/2.0 , ( p1.y + p2.y)/2.0 );

Line line1 , line2;

pointToLine( p1 , p2 , line1 );

perpendicularLine( line1 , mid );

mid = Point( ( p2.x + p3.x )/2.0 , ( p2.y + p3.y )/2.0 );

pointToLine( p2 , p3 , line2 );

perpendicularLine( line2 , mid );

if( !lineIntersection( line1 , line2 , c ) )

return false;

return true;

}

int main(){

Point p1 , p2 , p3;

while( scanf("%lf %lf %lf %lf %lf %lf" , &p1.x , &p1.y ,&p2.x, &p2.y

,&p3.x , &p3.y ) != EOF ){

Point c;

circle3Point( p1 , p2 , p3 ,c);

double r = dist( c , p1 );

printf("%.2lf\n" , r \* PI \* 2.0 );

}

return 0;

}

# Knight Moves

\*\*\*ID: 439

\*\*\*Tipo: Graphs theory, bfs

#include <bits/stdc++.h>

#include<stdio.h>

#include<queue>

#include<vector>

using namespace std;

struct Estado{

int x; //coordenada x

int y; //coordenada y

int d; //distancia recorrida

Estado(int x1,int y1,int d1):x(x1),y(y1),d(d1){};

};

int dx[ 8 ]={ -2 , -2 , -1 , -1 , 1 , 1 , 2 , 2 };

int dy[ 8 ]={ -1 , 1 , -2 , 2 , -2 , 2 , -1 , 1 };

const int h=8,w=8;

int tabla[h+3][w+3];

int bfs(int x, int y) {

Estado inicial (x,y,0); //posicion inicial

queue<Estado> cola; //cola que tendra estados de adyacentes

cola.push(inicial); //pongo el estado inicial

bool visitado[h+3][w+3];

memset(visitado,false,sizeof(visitado));

while(!cola.empty()){ //mientras no este vacia, mientras tengamos nodos para recorrer

Estado actual=cola.front(); //actual

cola.pop(); //elimino de la cola

if(tabla[actual.x][actual.y]==-2)return actual.d; //si encontramos salida retornarla

visitado[actual.x][actual.y]=true; //marco como visitado nodo actual

for(int i=0;i<8;i++){

int nx=actual.x+dx[i],ny=actual.y+dy[i];//vemos adyacentes

//si no excede limites y no es pared ademas de no haber sido visitado

if(nx>=0 && ny>=0 && nx<h && ny<w && !visitado[nx][ny]){

Estado ady(nx,ny,actual.d+1); //colocamos a la cola los adyacentes

cola.push(ady);

}

}

}

return 0;

}

int main(){

char a[3],b[3];

int xi,yi,xf,yf;

while(scanf("%s %s",&a,&b)!=EOF){

memset(tabla,0,sizeof(tabla));

xi=(a[0]-'a')+1;

yi=a[1]-'0';

xf=(b[0]-'a')+1;

yf=b[1]-'0';

tabla[xf-1][yf-1]=-2;

printf("To get from %s to %s takes %d knight moves.\n",a,b,bfs(xi-1,yi-1));

}

return 0;

}

# Eeny Meeny Moo

\*\*\*ID: 440

\*\*\*Tipo: Simulation, Josephus

#include <bits/stdc++.h>

//1 omitimos, empezamos desde 2 como si fuera 0

int survivor(int n, int m){

int i, s;

for (s = 0, i = 1; i <= n; i++) s = (s + m) % i;

return (s + 2);

}

int main(){

int n;

while( scanf("%d" , &n ) && n ){

for( int i = 1 ; ; ++i ){

if( survivor( n - 1 , i ) == 2 ){

printf("%d\n" , i );

break;

}

}

}

return 0;

}

# Lotto

\*\*\*ID: 441

\*\*\*Tipo: Backtracking

#include <bits/stdc++.h>

#include <vector>

using namespace std;

#define MAX 13

int k;

int a[ MAX ];

int seen[ MAX ];

vector<int> ady[ MAX ];

void dfs( int u , int len ){

seen[ u ] = 1;

if( len == 6 ){

bool b = false;

for( int i = 0 ; i < k ; ++i ){

if( seen[ i ] ){

if( b )printf(" ");

printf("%d" , a[ i ] );

b = true;

}

}

printf("\n");

return;

}

for( int v = 0 ; v < ady[ u ].size() ; ++v ){

int w = ady[ u ][ v ];

if( !seen[ w ] ){

dfs( w , len + 1 );

seen[ w ] = 0;

}

}

}

int main(){

bool b = false;

while( scanf("%d" , &k ) && k ){

if( b )putchar('\n');

b = true;

for( int i = 0 ; i < k ; ++i ){

scanf("%d" , &a[ i ] );

seen[ i ] = 0;

}

for( int i = 0 ; i < k ; ++i ){

for( int j = i + 1 ; j < k ; ++j ){

ady[ i ].push\_back( j );

}

}

for( int i = 0 ; i < k ; ++i ){

memset( seen , 0 , sizeof( seen ) );

dfs( i , 1 );

}

for( int i = 0 ; i < k ; ++i )ady[ i ].clear();

}

return 0;

}

# Humble Numbers

\*\*\*ID: 443

\*\*\*Tipo: Number Theory, dp

/\*

The 4011th humble number is 231525000.

The 412th humble number is 18000.

The 313th humble number is 7938.

\*/

#include <bits/stdc++.h>

#define MAX 5845

long long dp[ MAX ] = { 1 };

inline long long Min( long long a , long long b ){ return a < b ? a : b;}

void solve(){

int p1 = 0 , p2 = 0 , p3 = 0 , p4 = 0, mini, a = 2 , b = 3 , c = 5 , d = 7;

for( int i = 1 ; i < MAX ; ++i ){

dp[ i ] = Min( Min( dp[ p1 ] \* a , dp[ p2 ] \* b ) , Min( dp[ p3 ] \* c , dp[ p4 ] \* d ) );

if( dp[ i ] == dp[ p1 ] \* a ) p1++;

if( dp[ i ] == dp[ p2 ] \* b ) p2++;

if( dp[ i ] == dp[ p3 ] \* c ) p3++;

if( dp[ i ] == dp[ p4 ] \* d ) p4++;

}

}

const char \* getSuffix( int n ){

if( n % 10 == 1 && n % 100 != 11 ) return "st";

if( n % 10 == 2 && n % 100 != 12 ) return "nd";

if( n % 10 == 3 && n % 100 != 13 ) return "rd";

return "th";

}

int main(){

solve();

int n;

while( scanf("%d" , &n ) && n ){

printf("The %d%s humble number is %lld.\n" , n , getSuffix( n ) , dp[ n - 1 ] );

}

return 0;

}

# Kibbles `n' Bits `n' Bits `n' Bits

\*\*\*ID: 401

\*\*\*Tipo: Number Theory, Base conversion

#include <bits/stdc++.h>

#include<stdio.h>

#include<sstream>

using namespace std;

///DE A-F A ENTERO

int valor(char n){

int j,i;

for(i='A',j=10;i<='F' && j<=15;i++,j++){

if(n==i)break;

}

return j;

}

///DE ENTERO a A-F

char valor2(int n){

int i,j;

for(i=10,j='A';i<=15 && j<='F';i++,j++){

if(n==i)break;

}

return (char)j;

}

int pow(int a,int b){

int r=1;

for(int i=0;i<b;i++){

r\*=a;

}

return r;

}

int base(string n,int b){

int i,idx=0,aux;

int r=0;

for(i=n.length()-1;i>=0;i--){

if(isdigit(n[i])){

aux=n[i]-'0';

}

else{

aux=valor(n[i]);

}

if(aux>=b)return -1;

r+=(aux\*pow(b,idx));

idx++;

}

return r;

}

string toString(int n){

string s;

ostringstream buffer;

buffer<<n;

s= buffer.str();

return s;

}

string base2(int n,int b){

string str="";

int aux;

if(n==0)return "0";

while(n>=1){

aux=n%b;

if(aux>9 && aux<16){str=valor2(aux)+str;}

else {str=toString(aux)+str;}

n/=b;

}

return str;

}

string completar(string n){

string s="";

for(int i=0;i<13-n.length();i++){

s="0"+s;

}

return s+=n;

}

int main(){

int n,resp,num1,num2;

string n1,s,n2;

scanf("%d",&n);

for(int i=0;i<n;i++){

cin>>n1>>s>>n2;

num1=base(n1,16);

resp=num1;

n1=base2(num1,2);

n1=completar(n1);

num2=base(n2,16);

if(s.compare("+")==0)resp+=num2;

else resp-=num2;

n2=base2(num2,2);

n2=completar(n2);

cout<<n1<<" "<<s<<" "<<n2<<" = "<<resp<<endl;

}

return 0;

}

# Little Black Book

\*\*\*ID: 450

\*\*\*Tipo: Sorting

#include <bits/stdc++.h>

using namespace std;

struct Book{

string dep , title , fn , ln , hp , wp , cb , ha;

Book(){}

} v[ 100005 ];

bool cmp( Book b1 , Book b2 ){

return b1.ln < b2.ln;

}

int main(){

int t , l , q = 0;

char line[ 100005 ];

scanf("%d" , &t );

gets( line );

char dep[ 100005 ];

string s;

for( int qq = 0 ; qq < t ; ++qq ){

gets( dep );

while( gets( line ) ){

l = strlen( line );

if( l == 0 )break;

strcat( line , ",");

l++;

string ss[ 8 ];

for( int i = 0 , j = 0 ; i < l ; ++i ){

s = "";

while( i < l && line[ i ] != ',' ){

s += line[ i ];

i++;

}

ss[ j++ ] = s;

}

v[ q ].title = ss[ 0 ];

v[ q ].fn = ss[ 1 ];

v[ q ].ln = ss[ 2 ];

v[ q ].ha = ss[ 3 ];

v[ q ].hp = ss[ 4 ];

v[ q ].wp = ss[ 5 ];

v[ q ].cb = ss[ 6 ];

v[ q ].dep = string( dep );

q++;

}

}

sort( v , v + q , cmp );

for( int i = 0 ; i < q ; ++i ){

printf("----------------------------------------\n");

printf("%s %s %s\n" , v[ i ].title.c\_str() , v[ i ].fn.c\_str() , v[ i ].ln.c\_str() );

printf("%s\nDepartment: %s\n" , v[ i ].ha.c\_str() , v[ i ].dep.c\_str() );

printf("Home Phone: %s\nWork Phone: %s\nCampus Box: %s\n" , v[ i ].hp.c\_str() , v[ i ].wp.c\_str() , v[ i ].cb.c\_str() );

}

return 0;

}

# Project Scheduling

\*\*\*ID: 452

\*\*\*Tipo: Topological Sort

#include <bits/stdc++.h>

#include <cstdio>

#include <string>

#include <sstream>

#include <queue>

using namespace std;

#define MAX 30

char s[ MAX ];

int ady[ MAX ][ MAX ], w[ MAX ], degree[ MAX ], used[ MAX ];

int bfs(){

int dist[ MAX ];

memset( dist , 0 , sizeof( dist ) );

int ans = 0;

queue< int > Q;

for( int i = 0 ; i < MAX ; ++i ){

dist[ i ] = w[ i ];

if( used[ i ] && degree[ i ] == 0 ){

Q.push( i );

}

}

while( !Q.empty() ){

int act = Q.front(); Q.pop();

for( int i = 0 ; i < 28 ; ++i ){

if( ady[ act ][ i ] ){

degree[ i ]--;

dist[ i ] = max( dist[ i ] , w[ i ] + dist[ act ] ); //longest path DP

if( degree[ i ] == 0 )Q.push( i );

}

}

}

for( int j = 0 ; j < MAX ; ++j ){

if( !used[ j ] ) continue;

ans = max( ans , dist[ j ] );

}

return ans;

}

int main(){

char c;

int val , t;

scanf("%d" , &t );

gets( s );

gets( s );

for( int q = 0 ; q < t ; ++q ){

if( q )printf("\n");

memset( ady , 0 , sizeof( ady ) );

memset( w , 0 , sizeof( w ) );

memset( degree , 0 , sizeof( degree ) );

memset( used , 0 , sizeof( used ) );

while( gets( s ) ){

if( strlen( s ) == 0 ) break;

stringstream ss( s );

ss>>c>>val;

w[ c - 'A' ] = val;

used[ c - 'A' ] = 1;

if( ss>>s ){

for( int i = 0 ; s[ i ] ; ++i ){

degree[ c - 'A' ]++;

ady[ s[ i ] - 'A' ][ c - 'A' ] = 1;

used[ s[ i ] - 'A' ] = 1;

}

}

}

printf("%d\n" , bfs() );

}

return 0;

}

# Anagrams

\*\*\*ID: 454

\*\*\*Tipo: Strings

#include <bits/stdc++.h>

using namespace std;

static int cmpr(const void \*a, const void \*b) {

return strcmp(\*(char \*\*)a, \*(char \*\*)b);

}

int main(){

int t;

scanf("%d" , &t );

char s[ 105 ];

char \*words[ 105 ];

int len = 0;

gets( s );

gets( s );

int values[ 105 ][ 64 ];

for( int q = 0 ; q < t ; ++q ){

if( q )putchar('\n');

len = 0;

while( gets( s ) ){

if( strlen( s ) == 0 )break;

words[ len++ ] = strdup( s );

}

qsort( words , len , sizeof( char\* ) , cmpr );

memset( values , 0 , sizeof( values ) );

for( int i = 0 ; i < len ; ++i ){

int l = strlen( words[ i ] );

for( int j = 0 ; j < l ; ++j ){

if( words[ i ][ j ] == ' ') continue;

if( islower( words[ i ][ j ] ) )

values[ i ][ words[ i ][ j ] - 'a' ]++;

else values[ i ][ words[ i ][ j ] - 'A' + 26 ]++;

}

}

for( int i = 0 ; i < len ; ++i ){

if( i && !strcmp( words[ i ] , words[ i - 1 ] ) ) continue;

for( int j = i + 1 ; j < len ; ++j ){

if( !strcmp( words[ j ] , words[ j - 1 ] ) )continue;

int k;

for( k = 0 ; k < 64 ; ++k ){

if( values[ i ][ k ] != values[ j ][ k ] ) break;

}

if( k != 64 )continue;

printf("%s = %s\n" , words[ i ] , words[ j ] );

}

}

}

return 0;

}

# Periodic Strings

\*\*\*ID: 455

\*\*\*Tipo: KMP

#include <bits/stdc++.h>

using namespace std;

#define MAX 85

char s[ MAX ];

int len , b[ MAX ];

void preprocess(){

int i = 0 , j = -1; b[ 0 ] = -1;

while( i < len ){

while( j >= 0 && s[ i ] != s[ j ] ) j = b[ j ];

i++; j++;

b[ i ] = j;

}

}

int main(){

int t , periodic;

scanf("%d" , &t );

while( t-- > 0 && scanf("%s" , &s ) ){

len = strlen( s );

preprocess();

periodic = len - b[ len ];

if( len % periodic == 0 )

printf("%d\n" , periodic );

else printf("%d\n" , len );

if( t ) printf("\n");

}

return 0;

}

# Graph Connectivity

\*\*\*ID: 459

\*\*\*Tipo: Graph theory, union-find

#include <bits/stdc++.h>

#include<stdio.h>

#include<vector>

#include<set>

using namespace std;

#define MAX\_SIZE 30

///UNION-FIND

int parent[MAX\_SIZE],rank[MAX\_SIZE];

void Make\_Set(int x){

parent[x]=x;

rank[x]=0;

}

int Find(int x){

if(parent[x]!=x) parent[x]=Find(parent[x]);

return parent[x];

}

void Union(int PX, int PY){

if(rank[PX]>rank[PY]) parent[PY]=PX;

else{

parent[PX]=PY;

if(rank[PX]==rank[PY]) rank[PY]++;

}

}

void Merge(int x,int y){

Union(Find(x),Find(y));

}

///END UNION FIND

int main(){

int casos,nv,i,orig,dest,cont;

char tmp,C,input[20],\*p,f,s;

gets(input);

sscanf(input,"%d",&casos);

gets(input);

while(casos--){

gets(input);

f=input[0];

nv=f-'A';

for(i=0;i<=nv;i++){

Make\_Set(i);

}

while(gets(input)){

p=strtok(input," ");

if(p==NULL)break;

sscanf(p, "%c%c", &f, &s);

orig=(f-'A');

dest=(s-'A');

Merge(orig,dest);

}

int cnt = 0;

for( i = 0 ; i <= nv ; ++i ){

if( parent[ i ] == i ) cnt++;

}

printf("%d\n" , cnt );

if(casos>0) {

putchar('\n');

}

}

return 0;

}

# Overlapping Rectangles

\*\*\*ID: 460

\*\*\*Tipo: Math

#include<stdio.h>

int max(int a,int b){return (a>b)?a:b;}

int min(int a,int b){return (a<b)?a:b;}

int main(){

int test,n,xll1,xll2,yll1,yll2,xur1,xur2,yur1,yur2,it;

while(scanf("%d",&test)!=EOF){

for(it=0;it<test;it++){

scanf("%d %d %d %d",&xll1,&yll1,&xur1,&yur1);

scanf("%d %d %d %d",&xll2,&yll2,&xur2,&yur2);

xll1=max(xll1,xll2);

yll1=max(yll1,yll2);

xur1=min(xur1,xur2);

yur1=min(yur1,yur2);

if((xll1>=xur1) || (yll1>=yur1)){

printf("No Overlap\n");

}

else{

printf("%d %d %d %d\n",xll1,yll1,xur1,yur1);

}

if(it<test-1)printf("\n");

}

}

return 0;

}

# Bridge Hand Evaluator

\*\*\*ID: 462

\*\*\*Tipo: Simulation

#include <bits/stdc++.h>

using namespace std;

int value[ 256 ], stop[ 256 ];

int main(){

int tot , t, len;

char s[ 13 ][ 3 ];

while( scanf("%s" , &s[ 0 ] ) == 1 ){

tot = 0;

t = 12;

len = 0;

memset( value , 0 , sizeof( value ) );

memset( stop , 0 , sizeof( stop ) );

do{

if( s[ len ][ 0 ] == 'A') tot += 4;

else if( s[ len ][ 0 ] == 'K') tot += 3;

else if( s[ len ][ 0 ] == 'Q') tot += 2;

else if( s[ len ][ 0 ] == 'J') tot++;

value[ s[ len ][ 1 ] ]++;

if( t > 0 )scanf("%s" , &s[ ++len ] );

}while( t-- );

for( int i = 0 ; i < 13 ; ++i ){

if( s[ i ][ 0 ] == 'K' && value[ s[ i ][ 1 ] ] <= 1 ) tot--;

if( s[ i ][ 0 ] == 'Q' && value[ s[ i ][ 1 ] ] <= 2 ) tot--;

if( s[ i ][ 0 ] == 'J' && value[ s[ i ][ 1 ] ] <= 3 ) tot--;

if( s[ i ][ 0 ] == 'A' ) stop[ s[ i ][ 1 ] ] = 1;

if( s[ i ][ 0 ] == 'K' && value[ s[ i ][ 1 ] ] >= 2 ) stop[ s[ i ][ 1 ] ] = 1;

if( s[ i ][ 0 ] == 'Q' && value[ s[ i ][ 1 ] ] >= 3 ) stop[ s[ i ][ 1 ] ] = 1;

}

int tot2 = tot;

for( int i = 0 ; i < 4 ; ++ i ){

if( value[ "SHDC"[ i ] ] == 2 ) tot2++;

if( value[ "SHDC"[ i ] ] <= 1 ) tot2 += 2;

}

int idx = 'S';

for( int i = 1 ; i < 4 ; ++i ){

if( value[ "SHDC"[ i ] ] > value[ idx ] ) idx = "SHDC"[ i ];

}

if( tot >= 16 && stop[ 'S' ] && stop[ 'H' ] && stop[ 'D' ] && stop[ 'C' ])puts("BID NO-TRUMP");

else if( tot2 < 14 ) puts("PASS");

else printf("BID %c\n" , idx );

}

return 0;

}

# Mirror, Mirror

\*\*\*ID: 466

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

#define MAX 12

char ini[MAX][MAX];

char fin[MAX][MAX];

int n;

char aux[MAX][MAX];

char cini[MAX][MAX];

bool iguales(){

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

if(ini[i][j]!=fin[i][j])return false;

}

}

return true;

}

void rotar(){

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

aux[j][n-i-1]=ini[i][j];

}

}

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

ini[i][j]=aux[i][j];

}

}

}

void Reflex(){

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

aux[ n - j - 1 ][ i ] = ini[ j ][ i ];

}

}

for(int i=0;i<n;i++){

for(int j=0;j<n;j++){

ini[i][j]=aux[i][j];

}

}

}

int main(){

int q = 1;

while( cin>>n ){

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

cout<<"Pattern " <<q++<<" was ";

if(iguales()){cout<<"preserved."<<endl; continue;}

///rotamos 90 grados

rotar();

if(iguales()){cout<<"rotated 90 degrees."<<endl;continue;}

///rotamos 180 grados

rotar();

if(iguales()){cout<<"rotated 180 degrees."<<endl;continue;}

///rotamos 270 grados

rotar();

if(iguales()){cout<<"rotated 270 degrees."<<endl;continue;}

///reflexion

rotar();

Reflex();

if(iguales()){cout<<"reflected vertically."<<endl;continue;}

///combinacion

rotar();if(iguales()){cout<<"reflected vertically and rotated 90 degrees."<<endl; continue;}

rotar();if(iguales()){cout<<"reflected vertically and rotated 180 degrees."<<endl;continue;}

rotar();if(iguales()){cout<<"reflected vertically and rotated 270 degrees."<<endl; continue;}

///nada

rotar();if(iguales()){cout<<"preserved"<<endl; continue;}

///invalido

cout<<"improperly transformed."<<endl;

}

return 0;

}

# Magic Numbers

\*\*\*ID: 471

\*\*\*Tipo: Brute Force

#include <bits/stdc++.h>

bool unique( long long n ){

bool seen[ 10 ] = { 0 };

while( n > 0 ){

if( seen[ n % 10 ] )return false;

seen[ n % 10 ] = 1;

n /= 10;

}

return true;

}

int main(){

long long n , s2 , s1;

int t;

scanf("%d" , &t );

for( int q = 0 ; q < t ; ++q ){

if( q )putchar('\n');

scanf("%lld" , &n );

s2 = 1;

for( ; ; ++s2 ){

s1 = s2 \* n;

if( s1 > 9876543210LL ) break;

if( unique( s1 ) && unique( s2 ) ){

printf("%lld / %lld = %lld\n" , s1 , s2 , n );

}

}

}

return 0;

}

# Heads / Tails Probability

\*\*\*ID: 474

\*\*\*Tipo: Impresion

#include <bits/stdc++.h>

using namespace std;

#define MAX 1000005

double b[ MAX ];

int main(){

int n , i , digit;

b[ 0 ] = 1;

for( i = 1 ; i < MAX ; ++i ){

b[ i ] = b[ i - 1 ]/2.0;

if( b[ i ] < 1 ) b[ i ] \*= 10.0;

}

b[ 6 ] = 1.562;

while( scanf("%d" , &n ) != EOF ){

digit = floor( n \* log10( 2 ) ) + 1;

printf("2^-%d = %.3lfe-%d\n" , n , b[ n ] , digit );

}

return 0;

}

# Point in Figures Rectangles

\*\*\*ID: 476

\*\*\*Tipo: Geometry, point in rectangle.

#include <bits/stdc++.h>

#include<stdio.h>

#include<math.h>

#include<vector>

using namespace std;

#define EPS 1e-9

struct Point{

Point(double x1,double y1){

x=x1;

y=y1;

}

Point(){

}

double x;

double y;

///PRODUCTO PUNTO

double operator \*(const Point &p1)const{

return x\*p1.x+y\*p1.y;

}

///PRODUCTO CRUZ

double operator ^(const Point &p1)const{

return x\*p1.y-y\*p1.x;

}

Point operator-(const Point &p1)const{

Point a(x-p1.x,y-p1.y);

return a;

}

};

struct Rectangle{

Rectangle(Point ll,Point rr):leftT(ll),rightD(rr){}

Rectangle(Point lt,Point ld,Point rt,Point rd):leftT(lt),rightD(rd),leftD(ld),rightT(rt){}

Point leftD;

Point rightT;

Point leftT;

Point rightD;

};

///SI ESTA DENTRO DE RECTANGULO

///p: Punto averiguar si esta dentro de rectangulo

///L: botton-left de rectangulo

///R: top-right de rectangulo

bool insideRect(Point p,Point L, Point R){

if(p.x<R.x && L.x<p.x && p.y<R.y && L.y<p.y){

return true;

}

return false;

}

int main(){

double x1,y1,x2,y2,xx,yy;

char c[2];

vector<Rectangle> v;

while(scanf("%s",&c) ){

if(c[0]=='\*')break;

scanf("%lf %lf %lf %lf",&x1,&y1,&x2,&y2);

Point A(x1,y2),B(x2,y1);

Rectangle R(A,B);

v.push\_back(R);

}

int cont=1;

while(scanf("%lf %lf",&xx,&yy) && (xx!=9999.9 && yy!=9999.9)){

bool b=false;

Point P(xx,yy);

for(int i=0;i<v.size();i++){

if(insideRect(P,v[i].leftT,v[i].rightD)){

printf("Point %d is contained in figure %d\n",cont,(i+1));

b=true;

}

}

if(!b)printf("Point %d is not contained in any figure\n",cont);

cont++;

}

return 0;

}

# Point in Figures Rectangles and Circles

\*\*\*ID: 477

\*\*\*Tipo: Geometry, point in rectangle adn circle.

#include <bits/stdc++.h>

#include<stdio.h>

#include<vector>

#include<math.h>

using namespace std;

#define EPS 1e-9

struct Point{

Point(double x1,double y1){

x=x1;

y=y1;

}

Point(){

}

double x;

double y;

///PRODUCTO PUNTO

double operator \*(const Point &p1)const{

return x\*p1.x+y\*p1.y;

}

///PRODUCTO CRUZ

double operator ^(const Point &p1)const{

return x\*p1.y-y\*p1.x;

}

Point operator-(const Point &p1)const{

Point a(x-p1.x,y-p1.y);

return a;

}

};

struct Rectangle{

Rectangle(Point ll,Point rr):leftT(ll),rightD(rr){}

Rectangle(Point lt,Point ld,Point rt,Point rd):leftT(lt),rightD(rd),leftD(ld),rightT(rt){}

Point leftD;

Point rightT;

Point leftT;

Point rightD;

};

///SI ESTA DENTRO DE RECTANGULO

///p: Punto averiguar si esta dentro de rectangulo

///L: botton-left de rectangulo

///R: top-right de rectangulo

bool insideRect(Point p,Point L, Point R){

if(p.x<R.x && L.x<p.x && p.y<R.y && L.y<p.y){

return true;

}

return false;

}

///Distancia entre dos puntos

double dist(Point p1,Point p2){

return (p1.x-p2.x)\*(p1.x-p2.x)+(p1.y-p2.y)\*(p1.y-p2.y);

}

struct Figure{

int pos;

double radio;

Figure(int p,double r):pos(p),radio(r){}

Figure(){}

};

int main(){

double x1,y1,x2,y2,xx,yy,xc,yc,radio;

char c[2];

vector<pair<Rectangle,int> > v;

vector<pair<Point,Figure> > circle;

int pos=1;

while(scanf("%s",&c) ){

if(c[0]=='\*')break;

if(c[0]=='r'){

scanf("%lf %lf %lf %lf",&x1,&y1,&x2,&y2);

Point A(x1,y2),B(x2,y1);

Rectangle R(A,B);

v.push\_back(make\_pair(R,pos++));

}

else if(c[0]=='c'){

scanf("%lf %lf %lf",&xc,&yc,&radio);

Point A(xc,yc);

Figure F(pos++,radio);

circle.push\_back(make\_pair(A,F));

}

}

int cont=1;

while(scanf("%lf %lf",&xx,&yy) && (xx!=9999.9 && yy!=9999.9)){

bool b=false;

Point P(xx,yy);

for(int i=0;i<v.size();i++){

if(insideRect(P,v[i].first.leftT,v[i].first.rightD)){

printf("Point %d is contained in figure %d\n",cont,v[i].second);//(i+1));

b=true;

}

}

for(int i=0;i<circle.size();i++){

Point B(circle[i].first.x,circle[i].first.y);

if(sqrt(dist(P,B))<circle[i].second.radio){

printf("Point %d is contained in figure %d\n",cont,circle[i].second.pos);

b=true;

}

}

if(!b)printf("Point %d is not contained in any figure\n",cont);

cont++;

}

return 0;

}

# Point in Figures Rectangles,Circles and Triangles

\*\*\*ID: 478

\*\*\*Tipo: Geometry, point in figures.

#include <bits/stdc++.h>

#include<stdio.h>

#include<vector>

#include<math.h>

using namespace std;

#define EPS 1e-9

struct Point{

Point(double x1,double y1){

x=x1;

y=y1;

}

Point(){

}

double x;

double y;

///PRODUCTO PUNTO

double operator \*(const Point &p1)const{

return x\*p1.x+y\*p1.y;

}

///PRODUCTO CRUZ

double operator ^(const Point &p1)const{

return x\*p1.y-y\*p1.x;

}

Point operator-(const Point &p1)const{

Point a(x-p1.x,y-p1.y);

return a;

}

};

struct Rectangle{

Rectangle(Point ll,Point rr):leftT(ll),rightD(rr){}

Rectangle(Point lt,Point ld,Point rt,Point rd):leftT(lt),rightD(rd),leftD(ld),rightT(rt){}

Point leftD;

Point rightT;

Point leftT;

Point rightD;

};

///SI ESTA DENTRO DE RECTANGULO

///p: Punto averiguar si esta dentro de rectangulo

///L: botton-left de rectangulo

///R: top-right de rectangulo

bool insideRect(Point p,Point L, Point R){

if(p.x<R.x && L.x<p.x && p.y<R.y && L.y<p.y){

return true;

}

return false;

}

///Distancia entre dos puntos

double dist(Point p1,Point p2){

return (p1.x-p2.x)\*(p1.x-p2.x)+(p1.y-p2.y)\*(p1.y-p2.y);

}

/\*///Area triangulo

double areaT(Point A,Point B,Point C){

return (A.x - C.x)\*(B.y - C.y)-(A.y - C.y)\*(B.x - C.x);

}\*/

double areaT(Point A,Point B,Point C){

return ((C-A)^(B-A));

}

///SI ESTA DENTRO DE TRIANGULO

bool insideTriangle(Point A,Point B,Point C,Point P){

if(areaT(A,B,C)>0){

return areaT(A, B, P) > 0 && areaT(B, C, P) > 0 && areaT(C, A, P) > 0;

}

else return areaT(A, B, P) < 0 && areaT(B, C, P) < 0 && areaT(C, A, P) < 0;

}

struct Triangle{

Point A;

Point B;

Point C;

Triangle(Point a,Point b, Point c):A(a),B(b),C(c){}

};

struct Figure{

int pos;

double radio;

Figure(int p,double r):pos(p),radio(r){}

Figure(){}

};

int main(){

double x1,y1,x2,y2,xx,yy,xc,yc,radio,x3,y3;

char c[2];

vector<pair<Rectangle,int> > v;

vector<pair<Point,Figure> > circle;

vector<pair<Triangle,int> > triangle;

int pos=1;

while(scanf("%s",&c) ){

if(c[0]=='\*')break;

if(c[0]=='r'){

scanf("%lf %lf %lf %lf",&x1,&y1,&x2,&y2);

Point A(x1,y2),B(x2,y1);

Rectangle R(A,B);

v.push\_back(make\_pair(R,pos++));

}

else if(c[0]=='c'){

scanf("%lf %lf %lf",&xc,&yc,&radio);

Point A(xc,yc);

Figure F(pos++,radio);

circle.push\_back(make\_pair(A,F));

}

else if(c[0]=='t'){

scanf("%lf %lf %lf %lf %lf %lf",&x1,&y1,&x2,&y2,&x3,&y3);

Point A(x1,y1),B(x2,y2),C(x3,y3);

Triangle T(A,B,C);

triangle.push\_back(make\_pair(T,pos++));

}

}

int cont=1;

while(scanf("%lf %lf",&xx,&yy) && (xx!=9999.9 && yy!=9999.9)){

bool b=false;

Point P(xx,yy);

for(int i=0;i<v.size();i++){

if(insideRect(P,v[i].first.leftT,v[i].first.rightD)){

printf("Point %d is contained in figure %d\n",cont,v[i].second);//(i+1));

b=true;

}

}

for(int i=0;i<circle.size();i++){

Point B(circle[i].first.x,circle[i].first.y);

if(sqrt(dist(P,B))<circle[i].second.radio){

printf("Point %d is contained in figure %d\n",cont,circle[i].second.pos);

b=true;

}

}

for(int i=0;i<triangle.size();i++){

if(insideTriangle(triangle[i].first.A,triangle[i].first.B,triangle[i].first.C,P)){

printf("Point %d is contained in figure %d\n",cont,triangle[i].second);

b=true;

}

}

if(!b)printf("Point %d is not contained in any figure\n",cont);

cont++;

}

return 0;

}

# Permutation Arrays

\*\*\*ID: 482

\*\*\*Tipo: Ad hoc

#include <iostream>

#include <cstdlib>

#include <bits/stdc++.h>

#include <string>

#define N 1001

using namespace std;

int pos[N];

string array[N];

int main(){

int num\_of\_cases, n, temp;

cin>>num\_of\_cases;

for(int counter=0; counter<num\_of\_cases; counter++) {

if(counter!=0)

cout<<endl;

n=0;

do{

cin>>temp;

pos[temp-1]=n++;

}

while(getchar()!='\n');

for(int i=0;i<n;i++)

cin>>array[i];

for(int i=0;i<n;i++)

cout<<array[pos[i]]<<endl;

}

return 0;

}

# The Department of Redundancy Department

\*\*\*ID: 484

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

int main(){

string line;

int val;

unordered\_map<int,int> \_hash;

\_hash.clear();

vector<int> order;

while( cin>>val ){

if( \_hash.find( val ) == \_hash.end() ){

order.push\_back(val);

}

\_hash[val]++;

}

for( int i = 0 ; i < order.size() ; ++i )

printf("%d %d\n", order[i], \_hash[order[i]]);

return 0;

}

# Boggle Blitz

\*\*\*ID: 487

\*\*\*Tipo: Backtracking, DFS

#include <bits/stdc++.h>

using namespace std;

#define MAX 21

char ady[ MAX ][ MAX ];

vector<string> v;

int n;

bool seen[ MAX ][ MAX ];

int dx[ 8 ] = { 1 , -1 , 0 , 0 , 1 , 1 , -1 , -1 };

int dy[ 8 ] = { 0 , 0 , 1 , -1 , 1 , -1 , 1 , -1};

string s;

void dfs( int x , int y , int len ){

seen[ x ][ y ] = 1;

s += ady[ x ][ y ];

if( len >= 3 ){

v.push\_back( s );

}

for( int i = 0 ; i < 8 ; ++i ){

int nx = dx[ i ] + x, ny = dy[ i ] + y;

if( nx >= 0 && ny >= 0 && nx < n && ny < n && !seen[ nx ][ ny ] && ady[ nx ][ ny ] > ady[ x ][ y ] ){

dfs( nx , ny , len + 1 );

seen[ nx ][ ny ] = 0;

s = s.substr( 0 , s.length() - 1 );

}

}

}

bool cmp( string s1 , string s2 ){

if( s1.length() == s2.length() )return s1 < s2;

else{

if( s1.length() > s2.length()) return 0;

else return 1;

}

}

int main(){

int t;

scanf("%d" , &t );

for( int q = 0 ; q < t ; ++q ){

if( q )printf("\n");

scanf("%d" , &n );

for( int i = 0 ; i < n ; ++i )scanf("%s" , &ady[ i ] );

v.clear();

for( int i = 0 ; i < n ; ++i ){

for( int j = 0 ; j < n ; ++j ){

memset( seen , 0 , sizeof( seen ) );

s = "";

dfs( i , j , 1 );

}

}

if( v.size() > 0 ){

sort( v.begin() , v.end() , cmp );

string aux = v[ 0 ];

cout<<v[ 0 ]<<endl;

for( int i = 0 ; i < v.size(); ++i ){

if( v[ i ] != aux )cout<<v[ i ]<<endl;

aux = v[ i ];

}

}

}

return 0;

}

# Fibonacci Freeze

\*\*\*ID: 495

\*\*\*Tipo: Bignum, addition,Fibonacci.

#include <bits/stdc++.h>

using namespace std;

const int MAXD = 1040, DIG = 9, BASE = 1000000000;

const unsigned long long BOUND = numeric\_limits <unsigned long long> :: max () - (unsigned long long) BASE \* BASE;

struct bignum

{

int D, digits [MAXD / DIG + 2];

inline void trim ()

{

while (D > 1 && digits [D - 1] == 0)

D--;

}

inline void init (long long x)

{

memset (digits, 0, sizeof (digits));

D = 0;

do

{

digits [D++] = x % BASE;

x /= BASE;

}

while (x > 0);

}

inline bignum (long long x)

{

init (x);

}

inline bignum (int x = 0)

{

init (x);

}

inline char \*str ()

{

trim ();

char \*buf = new char [DIG \* D + 1];

int pos = 0, d = digits [D - 1];

do

{

buf [pos++] = d % 10 + '0';

d /= 10;

}

while (d > 0);

reverse (buf, buf + pos);

for (int i = D - 2; i >= 0; i--, pos += DIG)

for (int j = DIG - 1, t = digits [i]; j >= 0; j--)

{

buf [pos + j] = t % 10 + '0';

t /= 10;

}

buf [pos] = '\0';

return buf;

}

inline bignum operator + (const bignum &o) const

{

bignum sum = o;

int carry = 0;

for (sum.D = 0; sum.D < D || carry > 0; sum.D++)

{

sum.digits [sum.D] += (sum.D < D ? digits [sum.D] : 0) + carry;

if (sum.digits [sum.D] >= BASE)

{

sum.digits [sum.D] -= BASE;

carry = 1;

}

else

carry = 0;

}

sum.D = max (sum.D, o.D);

sum.trim ();

return sum;

}

};

bignum fib[5000];

int main(){

int n,i;

fib[0]=0;

fib[1]=1;

for(i=2;i<5001;i++){

fib[i]=fib[i-1]+fib[i-2];

}

while(scanf("%d",&n)!=EOF){

printf("The Fibonacci number for %d is %s\n",n,fib[n].str());

}

return 0;

}

# Simply Subsets

\*\*\*ID: 496

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

int main(){

int len\_a , len\_b , minl , x , p , n;

string s;

set<int> AA, BB;

set<int> :: iterator it2;

vector<int> A , B;

vector<int> :: iterator it;

while( getline( cin , s ) ){

len\_a = len\_b = 0;

stringstream ss( s + " ");

while( ss>>x ){

AA.insert( x );

}

getline( cin , s );

stringstream ss2( s + " ");

while( ss2>>x ){

BB.insert( x );

}

for( it2 = AA.begin() ; it2 != AA.end() ; ++it2 ){

A.push\_back( \*it2 );

}

for( it2 = BB.begin() ; it2 != BB.end() ; ++it2 ){

B.push\_back( \*it2 );

}

vector<int> C( A.size() + B.size() );

it = set\_intersection( A.begin() , A.end() , B.begin() , B.end() , C.begin() );

int intersection = ( int )( it - C.begin() );

if( intersection == 0 ) puts("A and B are disjoint");

else if( intersection == A.size() ){

if( intersection == B.size() ) puts("A equals B");

else if( B.size() > intersection ) puts( "A is a proper subset of B");

}

else if( intersection == B.size() ){

if( A.size() > intersection )puts("B is a proper subset of A");

}

else puts("I'm confused!");

A.clear();

B.clear();

AA.clear();

BB.clear();

}

return 0;

}

# Polly the Polynomial

\*\*\*ID: 498

\*\*\*Tipo: Math

#include <bits/stdc++.h>

using namespace std;

#define MAX 10005

int c[ MAX ] , x[ MAX ] , len , len\_x;

char s[ MAX ];

long long pot( int a , int b ){

long long r = 1;

for( int i = 1 ; i <= b ; ++i ) r \*= a;

return r;

}

long long evaluate( int idx ){

long long s = 0;

int i;

for( i = 0 ; i < len ; ++i ){

s += c[ i ] \* pot( x[ idx ] , len - i - 1 );

}

return s;

}

int main(){

int i;

while( gets( s ) ){

stringstream ss( s );

len = 0;

while( ss>>c[ len++ ] );

len--;

gets( s );

len\_x = 0;

ss.str( std::string() );

ss.clear();

ss<<s;

while( ss>>x[ len\_x++ ] );

len\_x--;

for( i = 0 ; i < len\_x ; ++i ){

if( i ) printf(" ");

printf("%lld" , evaluate( i ) );

}

printf("\n");

}

return 0;

}

# What's The Frequency, Kenneth?

\*\*\*ID: 449

\*\*\*Tipo: Ad hoc, Sorting

#include <bits/stdc++.h>

using namespace std;

#define MAX 5000

int main(){

char s[MAX];

int max, l;

while( gets ( s ) ){

int abc[60] = {0};

l=strlen( s );

for( int i = 0 ; i < l ; ++i){

if(isalpha( s[i] ) ){

abc[ s[i]-'A' ]++;

}

}

max = -1;

for( int i = 0 ; i < 60 ; ++i ){ if(abc[i] != 0 ) max = std::max( max , abc[i] ); }

for( int i = 0 ; i < 60 ; ++i ){

if( abc[i] == max ){

putchar((char)( i + 'A' ) );

}

}

if(max != -1)printf(" %d\n",max);

}

return 0;

}

# Black Box

\*\*\*ID: 501

\*\*\*Tipo: Data Structures

#include <iostream>

#include <cstdio>

#include <set>

#include <string>

#include <queue>

#include <map>

#include <stack>

using namespace std;

#define MAX 30005

int a[ MAX ] , u[ MAX ];

int main(){

int t , n , m;

scanf("%d" , &t );

for( int q = 0 ; q < t && scanf("%d %d" , &m , &n ) ; ++q ){

if( q )

printf("\n");

for( int i = 0 ; i < m && scanf("%d" , &a[i]); ++i );

for( int i = 0 ; i < n && scanf("%d" , &u[i]); ++i );

multiset<int> ms;

multiset<int>::iterator it;

for( int i = 0 , j = 0 ; i < m && j < n ; ++i ){

bool checked = false;

if( ms.size() >= 1 && it == ms.end() ){

it--;

ms.insert(a[i]);

if( \*it <= a[i] ){

it++;

}else

checked = true;

}else{

ms.insert( a[i] );

}

if( !checked && ms.size() > 1 && \*it > a[i] )

it--;

if( ms.size() == 1 )

it = ms.begin();

while( j < n && u[j] == ms.size() ){

printf("%d\n" , \*(it));

j++;

it++;

}

}

}

return 0;

}

# Jill Rides Again

\*\*\*ID: 507

\*\*\*Tipo: Maximum 1D sum

#include <bits/stdc++.h>

int main(){

int t , max, x , sum , idy ,n , l , bestlen;

scanf("%d",&t);

for( int q = 1 ; q <= t ; ++q ){

sum = 0;

max = 0;

idy = 0; l =0;

scanf("%d", &n);

for(int i = 0 ; i < n - 1; i++){

scanf("%d",&x);

sum+=x;

l++;

if( sum > max || ( sum == max && bestlen < l) ) { max = sum; idy = i + 1; bestlen = l;}

if( sum < 0) {sum = 0; l = 0; }

}

if( max > 0)printf("The nicest part of route %d is between stops %d and %d\n",q , idy - bestlen + 1 , idy + 1);

else printf("Route %d has no nice parts\n",q);

}

return 0;

}

# Rails

\*\*\*ID: 514

\*\*\*Tipo: Ad hoc, Stack

#include <bits/stdc++.h>

using namespace std;

int main( ){

int N, len;

bool first = false;

while( scanf("%d" , &N ) , N ){

int a[ N + 5 ], ini[ N + 5 ];

while( scanf("%d" , &a[ 0 ]) , a[ 0 ] ){

for( int i = 1 ; i < N ; ++i ){

scanf("%d" , &a[ i ] );

}

int i = 1 , idx = 0;

stack<int> S;

while( idx < N){

while( !S.empty() && a[ idx ] == S.top() ){

S.pop();

idx++;

}

if( idx == N || i > N ){ break; }

for(; i <= N ; ++i ){

if( i == a[ idx ] ){

idx++;

i++;

break;

}

else{

S.push( i );

}

}

}

if( idx == N )puts("Yes");

else puts("No");

}

first = true;

printf("\n");

}

return 0;

}

# Prime Ring Problem

\*\*\*ID: 524

\*\*\*Tipo: Backtracking, sieve, graph theory, DFS

#include<stdio.h>

#define MAX 17

int prime[33]={0, 0, 1, 1, 0, 1, 0, 1, 0, 0,

0, 1, 0, 1, 0, 0, 0, 1, 0, 1,

0, 0, 0, 1, 0, 0, 0, 0, 0, 1,

0, 1, 0};

int ady[MAX][MAX];

int visitado[MAX];

int resp[MAX];

int idx;

void dfs(int inicio,int tamano){

visitado[inicio]=true;

resp[idx++]=inicio;

if(idx==tamano && prime[resp[0]+resp[tamano-1]]){

putchar('1');

for(int i=1;i<tamano;i++){

printf(" %d",resp[i]);

}

putchar('\n');

}

for(int i=1;i<=tamano;++i){

if(ady[inicio][i]==1 && !visitado[i]){

dfs(i,tamano);

}

}

resp[--idx]=0;

visitado[inicio]=false;

}

void solve(int inicio,int final){

idx=0;

resp[idx++]=inicio;

visitado[inicio]=true;

for(int i=1;i<=final;i++){

if(ady[inicio][i]==1 && !visitado[i]){

dfs(i,final);

}

}

}

int main(){

int n,i,j,cont=1;

for(i=1;i<=MAX;i++){

for(j=1;j<=MAX;j++){

if(prime[i+j] && i!=j){ady[i][j]=1;}

}

}

while(scanf("%d",&n)!=EOF){

if(cont-1)putchar('\n');

printf("Case %d:\n",cont++);

if(!(n&1)){

for(i=0;i<=n;i++)visitado[i]=false;

solve(1,n);

}

}

return 0;

}

# String Distance and Transform Process

\*\*\*ID: 526

\*\*\*Tipo: Edit Distance - Print The Path

#include <bits/stdc++.h>

using namespace std;

#define MAX 90

#define INSERT 0

#define DELETE 1

#define CHANGE 2

#define MATCH 3

char A[ MAX ], B[ MAX ];

int lenA , lenB , dp[ MAX ][ MAX ] , prev[ MAX ][ MAX ];

void editDistance(){

lenA = strlen( A );

lenB = strlen( B );

int i , j;

for( i = 0 ; i <= lenA ; ++i ){

dp[ i ][ 0 ] = i;

prev[ i ][ 0 ] = DELETE;

}

for( i = 0 ; i <= lenB ; ++i ){

dp[ 0 ][ i ] = i;

prev[ 0 ][ i ] = INSERT;

}

for( i = 1 ; i <= lenA ; ++i ){

for( j = 1 ; j <= lenB ; ++j ){

if( A[ i - 1 ] == B[ j - 1 ] ){

dp[ i ][ j ] = dp[ i - 1 ][ j - 1 ];

prev[ i ][ j ] = MATCH;

}

else if( dp[ i - 1 ][ j ] > dp[ i ][ j - 1 ] && dp[ i - 1 ][ j - 1 ] > dp[ i ][ j - 1 ] ){

dp[ i ][ j ] = dp[ i ][ j - 1 ] + 1;

prev[ i ][ j ] = INSERT;

}

else if( dp[ i - 1 ][ j ] > dp[ i - 1 ][ j - 1 ]){

dp[ i ][ j ] = dp[ i - 1 ][ j - 1 ] + 1;

prev[ i ][ j ] = CHANGE;

}

else{

dp[ i ][ j ] = dp[ i - 1 ][ j ] + 1;

prev[ i ][ j ] = DELETE;

}

}

}

}

int len, k;

void printEditDistance( int i , int j ){

if( !i && !j ) return;

if( prev[ i ][ j ] == MATCH ){

printEditDistance( i - 1 , j - 1 );

return;

}

if( prev[ i ][ j ] == INSERT ){

printEditDistance( i , j - 1 );

printf("%d Insert %d,%c\n" , len++ , j , B[ j - 1 ] );

k--;

}

if( prev[ i ][ j ] == DELETE ){

printEditDistance( i - 1 , j );

printf("%d Delete %d\n" , len++ , i - k );

k++;

}

if( prev[ i ][ j ] == CHANGE ){

printEditDistance( i - 1 , j - 1 );

printf("%d Replace %d,%c\n" , len++ , j , B[ j - 1 ] );

}

}

int main(){

bool b = false;

while( gets( A ) ){

gets( B );

if( b ) printf("\n");

b = true;

editDistance();

printf("%d\n" , dp[ lenA ][ lenB ] );

len = 1;

k = 0;

printEditDistance( lenA , lenB );

}

return 0;

}

# Binomial Showdown

\*\*\*ID: 530

\*\*\*Tipo: Number theory, gcd, binomial theorem.

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

int main() {

int n, k;

unsigned long long r;

while(cin >> n >> k && ((n != 0) || (k != 0))) {

r = 1;

/\* Si k es mas de la mitad usamos complemento \*/

if(k > (n / 2)) {

k = n - k;

}

/\*

\* C(n,k) = n! / (k!(n-k)!) =

\* (n)(n-1)(...)(n-k+1) / 2\*3\*4\*(...)\*k

\*/

for (int i=0; i<k; i++) {

r = r \* (n - i); /\* (n)(n-1)(...)(n-k+1) \*/

r = r / (1 + i); /\* 2\*3\*4\*(...)\*k \*/

}

cout << r << endl;

}

}

# Compromise

\*\*\*ID: 531

\*\*\*Tipo: LCS print path

#include <bits/stdc++.h>

using namespace std;

#define MAXN 1005

#define MATCH 0

#define INSERT 1

#define DELETE 2

map< string , int > mp;

string v[ MAXN ];

char line[ MAXN ];

int A[ MAXN ] , B[ MAXN ] , lenA , lenB , dp[ MAXN ][ MAXN ] , prev[ MAXN ][ MAXN ];

int MAX( int a , int b , int c ){ return max( a , max( b , c ) ); }

void LCS(){

int i , j;

memset( prev , -1 , sizeof( prev ) );

for( i = 1 ; i <= lenA ; ++i ){

for( j = 1 ; j <= lenB ; ++j ){

if( A[ i - 1 ] == B [ j - 1 ] ){

dp[ i ][ j ] = dp[ i - 1 ][ j - 1 ] + 1;

prev[ i ][ j ] = MATCH;

}

else if( dp[ i - 1 ][ j ] > dp[ i ][ j - 1 ] ){

dp[ i ][ j ] = dp[ i - 1 ][ j ];

prev[ i ][ j ] = DELETE;

}

else{

dp[ i ][ j ] = dp[ i ][ j - 1 ];

prev[ i ][ j ] = INSERT;

}

}

}

}

bool first;

void printLCS( int i , int j ){

if( prev[ i ][ j ] == -1 ) return;

if( prev[ i ][ j ] == INSERT ){

printLCS( i , j - 1 );

}

if( prev[ i ][ j ] == DELETE ){

printLCS( i - 1 , j );

}

if( prev[ i ][ j ] == MATCH ){

printLCS( i - 1 , j - 1 );

if( !first ) printf(" ");

first = false;

printf("%s" , v[ A[ i - 1 ] ].c\_str() );

}

}

void solve(){

LCS();

first = true;

printLCS( lenA , lenB );

printf("\n");

}

int main(){

int len;

while( scanf("%s" , &line ) != EOF ){

len = lenA = lenB =0;

mp.clear();

if( line[ 0 ] != '#' ){

do{

if( !mp.count( line ) ){

A[ lenA ] = len;

v[ len ] = line;

mp[ line ] = len++;

}

else A[ lenA ] = mp[ line ];

lenA++;

}while( scanf("%s" , &line ) , line[ 0 ] != '#' );

}

scanf("%s" , &line );

if( line[ 0 ] != '#' ){

do{

if( !mp.count( line ) ){

B[ lenB ] = len;

v[ len ] = line;

mp[ line ] = len++;

}

else B[ lenB ] = mp[ line ];

lenB++;

}while( scanf("%s" , &line ) , line[ 0 ] != '#' );

}

solve();

}

}

# Dungeon Master

\*\*\*ID: 532

\*\*\*Tipo: Graph Theory, BFS

#include <bits/stdc++.h>

#include<stdio.h>

#include<vector>

#include<string>

#include<queue>

using namespace std;

#define MAX 30

struct Estado{

int x; //coordenada x

int y; //coordenada y

int z;

Estado(int x1=-1,int y1=-1,int z1=-1):x(x1),y(y1),z(z1){};

};

void bfs(vector< vector<string> > niveles,int h,int w, int x, int y, int z) {

Estado inicial(z,x,y);

queue<Estado> cola; //cola que tendra estados de adyacentes

cola.push(inicial); //pongo el estado inicial

int distance[niveles.size()][h][w];

memset(distance,-1,sizeof(distance));

distance[inicial.x][inicial.y][inicial.z]=0;

while(!cola.empty()){ //mientras no este vacia, mientras tengamos nodos para recorrer

Estado actual=cola.front(); //actual

cola.pop(); //elimino de la cola

if(niveles[actual.x][actual.y][actual.z]=='E'){

printf("Escaped in %d minute(s).\n",distance[actual.x][actual.y][actual.z]);

return;

}

int dx[] = { 0, 0, 0, 0,-1, 1};

int dy[] = { 0, 0,-1, 1, 0, 0};

int dz[] = {-1, 1, 0, 0, 0, 0};

for(int i=0;i<6;i++){

int nx=actual.x+dx[i],ny=actual.y+dy[i],nz=actual.z+dz[i];//vemos adyacentes

//si no excede limites y no es pared ademas de no haber sido visitado

if(nx>=0 && ny>=0 && ny<h && nz<w && nz>=0 && nx<niveles.size() && niveles[nx][ny][nz]!='#' && distance[nx][ny][nz] == -1){

Estado ady(nx,ny,nz); //colocamos a la cola los adyacentes

distance[nx][ny][nz] = distance[actual.x][actual.y][actual.z] + 1;

cola.push(ady);

}

}

}

printf("Trapped!\n");

}

int main(){

int l,f,c,i,j,k,tm,xi,yi,xf,yf,zi;

char line[50],C;

while(scanf("%d %d %d",&l,&f,&c) && l!=0 && f!=0 && c!=0){

vector< vector<string> > laberinto;

cin.get();

for(k=0;k<l;k++){

vector<string> nivel;

for(i=0;i<f;i++){

scanf("%s",&line);

for(j=0;j<c;j++){

if(line[j]=='S'){xi=i;yi=j;zi=k;}

}

nivel.push\_back(line);

}

laberinto.push\_back(nivel);

}

bfs(laberinto,f,c,xi,yi,zi);

}

return 0;

}

# Tree Recovery

\*\*\*ID: 536

\*\*\*Tipo: Recursion

#include <bits/stdc++.h>

#include <string.h>

#define MAX 30

int x;

void solve( char \*s1 , char \*s2 , int l ){

if( l < 1 )return;

if( l == 1 ){

putchar( s1[ 0 ] );

return;

}

int i;

for( i = 0 ; i < l ; ++i ){

if( s1[ 0 ] == s2 [ i ] ){ break;}

}

solve( s1 + 1 , s2 , i);

solve( s1 + 1 + i , s2 + i + 1 , l - i - 1);

putchar( s1 [ 0 ] );

}

int main(){

char s1[ MAX ], s2[ MAX ];

int l1, l2 ;

while( scanf("%s %s", s1 , s2 ) != EOF ){

l1 = strlen( s1 );

solve( s1 , s2 , l1);

putchar('\n');

}

return 0;

}

# The Settlers of Catan

\*\*\*ID: 539

\*\*\*Tipo: Backtracking

#include <bits/stdc++.h>

#define MAX 26

int ady[ MAX ][ MAX ];

int V;

int resp;

void dfs( int n , int len ){

if( resp < len )resp = len;

for( int i = 0 ; i < V ; ++i ){

if( ady[ n ][ i ] ){

ady[ n ][ i ] = ady[ i ][ n ] = 0;

dfs( i , len + 1 );

ady[ n ][ i ] = ady[ i ][ n ] = 1;

}

}

}

int main(){

int E, u , v;

while( scanf("%d %d" , &V , &E ) && V|E ){

memset( ady , 0 , sizeof( ady ) );

while( E-- ){

scanf("%d %d" , &u , &v );

ady[ u ][ v ] = ady[ v ][ u ] = 1;

}

resp = 0;

for( int i = 0 ; i < V ; ++i ){

dfs( i , 0 );

}

printf("%d\n" , resp );

}

return 0;

}

# Team Queue

\*\*\*ID: 540

\*\*\*Tipo: Queue

#include <bits/stdc++.h>

using namespace std;

#define MAX 200005

#define MTEAM 1000005

int team[ MTEAM ];

int pos[ 10005 ];

int head, sz, n;

int ini;

struct Queue{

int top, sz , a[ 1005 ];

}Q[ 1005 ];

int lista[ 1005 ]; // tengo un arreglo de colas, cada vez q se elimine una cola, crearemos una nueva, no eliminaremos

void init(){

head = sz = 0;

for( int i = 0 ; i < n ; ++i ) Q[ i ].top = Q[ i ].sz = 0;

}

void Push( int x ){

int idx = pos[ team[ x ] ];

if( Q[ idx ].top == Q[ idx ].sz ){//si vacio

Q[ idx ].a[ 0 ] = x; //creo nueva cola dentro de indice dado

Q[ idx ].sz++;

lista[ sz++ ] = idx;

}

else {

Q[ idx ].a[ Q[ idx ].sz++ ] = x; //si existe elemento aumento tamano de la iesima cola

}

}

int Pop(){

int ini = lista[ head ]; //la 1era cola

int x = Q[ ini ].a[ Q[ ini ].top++ ];

if( Q[ ini ].top == Q[ ini ].sz ){

head++;

pos[ team[ x ] ] = -1; //hallo nueva posicion de cola en lista

}

return x;

}

int main(){

int t, x , len, q = 1;

char s[ 20 ];

while( scanf("%d" , &n ) , n ){

memset( pos , -1 , sizeof( pos ) );

for( int i = 0 ; i < n ; ++i ){

scanf("%d" , &t );

for( int j = 0 ; j < t ; ++j ){

scanf("%d" , &x );

team[ x ] = i;

}

}

init();

printf("Scenario #%d\n" , q++ );

len = 0;

while( scanf("%s" , &s ) && strcmp( s , "STOP") ){

if( !strcmp( s , "ENQUEUE")){

scanf("%d" , &x );

if( pos[ team[ x ] ] == -1 ){

pos[ team[ x ] ] = len++;

}

Push( x );

}

else{

printf("%d\n" , Pop() );

}

}

printf("\n");

}

return 0;

}

# Error Correction

\*\*\*ID: 501

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

int main(){

int n,i,j,tabla[102][102],parf,imparf,sumf,sumc,imparc,parc,idxf,idxc;

while(scanf("%d",&n) && n!=0){

parf=0;

imparf=0;

parc=0;

imparc=0;

sumf=0;

sumc=0;

for(i=0;i<n;i++){

for(j=0;j<n;j++){

scanf("%d",&tabla[i][j]);

}

}

for(i=0;i<n;i++){

if(imparf>1 || imparc>1)break;

for(j=0;j<n;j++){

if(tabla[j][i]==1){sumc++;}

if(tabla[i][j]==1){sumf++;}

}

if(sumc%2==0)parc++;

else {imparc++;idxc=i;}

sumc=0;

if(sumf%2==0)parf++;

else {imparf++;idxf=i;}

sumf=0;

}

if(imparf>1 || imparc>1)printf("Corrupt\n");

else if(imparf==0 && imparc==0) printf("OK\n");

else {

printf("Change bit (%d,%d)\n",idxf+1,idxc+1);

}

}

return 0;

}

# Goldbach's Conjecture

\*\*\*ID: 543

\*\*\*Tipo: Number Theory, sieve

#include<stdio.h>

#define MAX 1000005

bool primes[MAX];

void sieve(){

primes[0]=false;

for(int i=1;i<MAX;i++) primes[i]=true;

for(int i=2;i\*i<=MAX;i++){

if(primes[i]){

for(int j=i;j\*i<=MAX;j++){

primes[j\*i]=false;

}

}

}

}

int main(){

int n,i,j,resp;

sieve();

while(scanf("%d",&n) && n!=0){

resp=0;

for(i=2;i<=n/2;i++){

if(!primes[i])continue;

if(primes[i] && primes[n-i]){printf("%d = %d + %d\n",n,i,n-i);break;}

}

}

return 0;

}

/\*

int table[1000000];

void criba(int max){

table[0] = false;

table[1] = false;

int i,h;

for(i = 2; i <= max; ++i){

if(!i&1){

table[i]=0;

}

else{

table[i] = i;

}

}

for(i = 2; i\*i <= max; ++i)

if(table[i])

for(h = 2; i\*h <= max; ++h)

table[i\*h] = 0;

}

int main(){

int n,cont,i,j;

criba(1000000);

while(scanf("%d",&n)){

if(n<6 || n>=1000000 || n==0){break;}

for(j=n-1;j>1;j--){

if(table[j]==0){continue;}

for(i=2;i<n;i++){

if(table[i]+table[j]==n){

cout<<n<<" = "<<table[i]<<" + "<<table[j]<<endl;

cont++;

break;

}

}

if(cont>0){break;}

}

if(cont==0){

cout<<"Goldbach's conjecture is wrong."<<endl;

}

cont=0;

}

return 0;

}

/\*

bool isPrime(int n){

int raiz=(int)sqrt((double)n);

for(int i=2;i<=raiz;i++){

if(n%i==0){

return 0;

}

}

return 1;

}

int main(){

int n,cont,i,j;

//criba(1000000);

while(scanf("%d",&n)){

if(n<6 || n>=1000000 || n==0){break;}

for(j=n-1;j>1;j--){

//if(table[j]==0){continue;}

if(j%2==0 || !isPrime(j)){continue;}

for(i=2;i<n;i++){

/\*if(table[i]+table[j]==n){

cout<<n<<" = "<<table[i]<<" + "<<table[j]<<endl;

cont++;

break;

}

if(i%2==0 || !isPrime(i)){continue;}

else{

if(i+j==n){

cout<<n<<" = "<<i<<" + "<<j<<endl;

cont++;

break;

}

}

}

if(cont>0){break;}

}

if(cont==0){

cout<<"Goldbach's conjecture is wrong."<<endl;

}

cont=0;

}

return 0;

}\*/

# Heavy Cargo

\*\*\*ID: 544

\*\*\*Tipo: Floyd Warshall, Maxmin

#include <bits/stdc++.h>

#include <iostream>

using namespace std;

#define MAX 205

#define INF 9999999

int ady[ MAX ][ MAX ];

void Init( int V ){

for( int i = 0 ; i < V ; ++i ){

for( int j = 0 ; j < V ; ++j ){

ady[ i ][ j ] = -1;

}

ady[ i ][ i ] = 0;

}

}

void Floyd( int V ){

for( int k = 0 ; k < V ; ++k ){

for( int i = 0 ; i < V ; ++i ){

for( int j = 0 ; j < V ; ++j ){

ady[ i ][ j ] = ady[ j ][ i ] = max( ady[ i ][ j ] , min( ady[ i ][ k ] , ady[ k ][ j ] ) );

}

}

}

}

char cities[ MAX ][ 40 ];

int numcities;

int get\_Id( char \*s ){

for( int i = 0 ; i < numcities ; ++i ){

if( strcmp( s , cities[ i ] ) == 0 ) return i;

}

strcpy( cities[ numcities ] ,s );

return numcities++;

}

int main(){

char u[ 40 ] , v [ 40 ];

int w, uu, vv , q = 1 ,V ,E;

while( scanf("%d %d", &V ,&E ) && V|E ){

Init( V );

numcities = 0;

while( E-- ){

scanf("%s %s %d" , &u , &v , &w );

uu = get\_Id( u ); vv = get\_Id( v );

ady[ uu ][ vv ] = ady[ vv ][ uu ] = w;

}

Floyd( V );

scanf("%s %s" ,&u ,&v );

printf("Scenario #%d\n%d tons\n\n" ,q++ , ady[ get\_Id( u ) ][ get\_Id( v ) ]);

memset( ady , 0 ,sizeof( ady ) );

}

return 0;

}

# Head

\*\*\*ID: 545

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

#define EPS 1e-9

#define MAX 9005

double b[ MAX ];

int main(){

int n , i , digit , t;

b[ 0 ] = 1;

for( i = 1 ; i < MAX ; ++i ){

b[ i ] = b[ i - 1 ]/2.0;

if( b[ i ] < 1 ) b[ i ] \*= 10.0;

}

scanf("%d" , &t );

while( t-- > 0 && scanf("%d" , &n ) ){

digit = floor( n \* log10( 2 ) ) + 1;

printf("2^-%d = %.3lfE-%d\n" , n , b[ n ] + EPS , digit );

}

return 0;

}

# Image Recognizer

\*\*\*ID: 546

\*\*\*Tipo: Simulation

#include <bits/stdc++.h>

using namespace std;

#define MAX 105

char s[ MAX ];

int row\_i , col\_i , I[ MAX ][ MAX ] , T[ MAX ][ MAX ] , row\_t , col\_t;

void solve(){

int i , j , ii , jj , maxi = -1<<30 , sum , x = 0 , y = 0;

for( i = 0 ; i + row\_t <= row\_i ; ++i ){

for( j = 0 ; j + col\_t <= col\_i ; ++j ){

sum = 0;

for( ii = 0 ; ii < row\_t ; ++ii ){

for( jj = 0 ; jj < col\_t ; ++jj ){

sum += I[ i + ii ][ j + jj ] \* T[ ii ][ jj ];

}

}

if( sum > maxi ){

maxi = sum;

x = i;

y = j;

}

}

}

printf("(%d,%d)\n" , y , x );

}

int main(){

gets( s );

while( strlen( s ) > 0 && s[ 0 ] != 'F' ){

row\_i = row\_t = 0;

while( gets( s ) && !isalpha( s[ 0 ] ) ){

if( strlen( s ) == 0 ) continue;

stringstream ss( s );

col\_i = 0;

while( ss>>I[ row\_i ][ col\_i ] ){ col\_i++; }

row\_i++;

}

while( gets( s ) && !isalpha( s[ 0 ] ) ){

if( strlen( s ) == 0 ) continue;

stringstream ss( s );

col\_t = 0;

while( ss>>T[ row\_t ][ col\_t ] ){ col\_t++; }

row\_t++;

}

solve();

}

return 0;

}

DDF

\*\*\*ID: 547

\*\*\*Tipo: Prime Factorization

#include <bits/stdc++.h>

using namespace std;

#define MAX 3100

int s[ MAX ];

int fact[ 80 ]; //MAX es la raiz cuadrada del maximo valor posible para n

int factores( int n ){

int len , l , i;

len = (int)( sqrt( n ) );

l = 0;

for( i = 1 ; i <= len ; ++i ){

if( n % i == 0 ){

fact[ l++ ] = i;

if( i != n/i )

fact[ l++ ] = n/i;

}

}

int sum = 0;

for( i = 0 ; i < l ; ++i ){

while( fact[ i ] > 0 ){

sum += fact[ i ] % 10;

fact[ i ] /= 10;

}

}

return sum;

}

bool seen[ MAX ];

vector< int > ady[ MAX ];

void gen(){

int i , j , n , len;

for( i = 0 ; i < MAX ; ++i ) s[ i ] = factores( i );

for( i = 0 ; i < MAX ; ++i ){

memset( seen , 0 , sizeof( seen ) );

for( j = i ; !seen[ j ] ; j = s[ j ] ){

seen[ j ] = 1;

ady[ i ].push\_back( j );

}

}

}

int main(){

gen();

int a , b , maxi , len , ini , q = 1 , i;

while( scanf("%d %d" , &a , &b ) != EOF ){

printf("Input%d: %d %d\n" , q , a , b );

if( b < a ) swap( a , b );

maxi = ini = -1;

for( ; a <= b ; ++a ){

if( (int)ady[ a ].size() > maxi ){

maxi = (int)ady[ a ].size();

ini = a;

}

}

printf("Output%d:" , q++ , ini );

for( i = 0 ; i < ady[ ini ].size() ; ++i ){

printf(" %d" , ady[ ini ][ i ] );

}

printf("\n");

}

return 0;

}

# Tree

\*\*\*ID: 548

\*\*\*Tipo: Tree Traversal

#include <bits/stdc++.h>

using namespace std;

#define MAX 100005

char s[ MAX ];

int in[ MAX ] , pos[ MAX ];

int mini , min\_vertex;

void solve( int l\_i , int len\_in , int l\_p , int len\_pos , int sum ){

if( len\_pos <= 0 ) return;

int root = pos[ len\_pos - 1 ] , i;

//si es hoja

if( l\_i + 1 == len\_in ){

if( sum + in[ l\_i ] < mini ){

mini = sum + in[ l\_i ];

min\_vertex = in[ l\_i ];

}

else if( sum + in[ l\_i ] == mini ){

min\_vertex = min( min\_vertex , in[ l\_i ] );

}

return;

}

for( i = l\_i ; i < len\_in ; ++i ){

if( in[ i ] == root ){

break;

}

}

if( i == len\_in ) return;

solve( l\_i , i , l\_p , l\_p + i - l\_i , sum + root );

solve( i + 1 , len\_in , len\_pos - ( len\_in - i ) , len\_pos - 1 , sum + root );

}

int main(){

int len\_in , len\_pos;

int len , i , j;

while( gets( s ) ){

len\_in = len\_pos = 0;

stringstream ss( s );

while( ss>>in[ len\_in ] )len\_in++;

gets( s );

stringstream ss2( s );

while( ss2>>pos[ len\_pos ] )len\_pos++;

mini = min\_vertex = 1<<30;

solve( 0 , len\_in , 0 , len\_pos , 0 );

printf("%d\n" , min\_vertex );

}

return 0;

}

# Multiplying by Rotation

\*\*\*ID: 550

\*\*\*Tipo: Math

#include <bits/stdc++.h>

int main(){

int base , n , m , i;

while( scanf("%d %d %d" , &base , &n , &m ) != EOF ){

if( m == 1 ) puts("1");

else{

int act = n , carry = 0;

for( i = 1 ; ; ++i ){

act = act \* m + carry;

carry = act / base;

act %= base;

if( carry == 0 && act == n ) break;

}

printf("%d\n" , i );

}

}

return 0;

}

# Nesting a Bunch of Brackets

\*\*\*ID: 551

\*\*\*Tipo: Stack

#include <bits/stdc++.h>

using namespace std;

struct Data{

int idx, p;

Data( int pp , int id ):p( pp ), idx( id ){}

};

//(\* 1, { 2 ( 3 [ 4

int main(){

char s[ 3005 ];

int l;

while( gets( s ) ){

l = strlen( s );

stack< Data > S;

int ans = -1;

for( int i = 0 , j = 0 ; i < l ; ++i , ++j ){

if( s[ i ] == '(' ){

if( i + 1 < l && s[ i + 1 ] == '\*' ){

S.push( Data( 1 , j + 1 ) ); i++;

}

else{

S.push( Data( 3 , j + 1 ) );

}

}

else if( s[ i ] == '{') S.push( Data( 2 , j + 1 ) );

else if( s[ i ] == '[') S.push( Data( 4 , j + 1 ) );

else if( s[ i ] == '}'){

if( S.empty() ){

ans = i + 1;

break;

}

else{

if( S.top().p != 2 ){

ans = S.top().idx + 1;

break;

}

S.pop();

}

}

else if( s[ i ] == ']' ){

if( S.empty() ){

ans = j + 1;

break;

}

else{

if( S.top().p != 4 ){

ans = S.top().idx;

break;

}

S.pop();

}

}

else if( s[ i ] == '\*' ){

if( i + 1 < l && s[ i + 1 ] == ')'){

if( S.empty() ){

ans = i + 1;

break;

}

else{

if( S.top().p != 1 ){

ans = S.top().idx;

break;

}

S.pop();

}

}

}

else if( s[ i ] == ')'){

if( S.empty() ){

ans = i + 1;

break;

}

else{

if( S.top().p != 2 ){

ans = S.top().idx;

break;

}

S.pop();

}

}

}

if( ans == -1 ) puts("YES");

else printf("NO %d\n" , ans );

}

return 0;

}

# Filling the Gaps

\*\*\*ID: 552

\*\*\*Tipo: Trie

#include <bits/stdc++.h>

using namespace std;

#define MAX 2 //alfabeto

struct Node{

int words;

Node \*hijo[ MAX ];

Node(){

words = 0;

for( int i = 0 ; i < MAX ; ++i ) hijo[ i ] = NULL;

}

};

class Trie{

public:

Trie(){

root = new Node();

}

void printAllWords( Node \*raiz , string s );

void insert( Node \*raiz , char \*s , int l );

int countAllWords( Node \*raiz );

Node\* getRoot();

bool contains( char \*s );

private:

Node\* root;

};

Node\* Trie::getRoot(){

return root;

}

void Trie::insert( Node \*t , char \*s , int l ){

if( s[ l ] == '\0' ){

t -> words = 1;

return;

}

if( s[ l ] == '\*' ){

if( t -> hijo[ 0 ] == NULL ) t -> hijo[ 0 ] = new Node();

insert( t -> hijo[ 0 ] , s , l + 1 );

if( t -> hijo[ 1 ] == NULL ) t -> hijo[ 1 ] = new Node();

insert( t -> hijo[ 1 ] , s , l + 1 );

}

else{

if( t -> hijo[ s[ l ] - '0' ] == NULL ) t -> hijo[ s[ l ] - '0' ] = new Node();

insert( t -> hijo[ s[ l ] - '0' ] , s , l + 1 );

}

}

void Trie::printAllWords( Node \*t, string word = ""){

if( t -> words > 0 ){

printf("%s\n" , word.c\_str());

}

int i;

for( i = 0 ; i < MAX ; ++i ){

if( t -> hijo[ i ] != NULL ) printAllWords( t -> hijo[ i ] , word + ( char )( i + '0' ) );

}

}

int Trie::countAllWords( Node \*t ){

int ans = 0;

if( t -> words > 0 ){

ans++;

}

int i;

for( i = 0 ; i < MAX ; ++i ){

if( t -> hijo[ i ] != NULL ) ans += countAllWords( t -> hijo[ i ] );

}

return ans;

}

bool Trie::contains( char \*s ){

int i;

Node \*t = root;

for( i = 0 ; s[ i ] ; ++i ){

if( t -> hijo[ s[ i ] - '0' ] == NULL ) return false;

t = t -> hijo[ s[ i ] - '0' ];

}

return t -> words > 0;

}

#define MAXN 16

char s[ 1005 ][ MAXN ] , str[ MAXN ];

int main(){

int len , n , i , j , sum;

while( scanf("%d %d" , &len , &n ) , len | n ){

Trie \*trie = new Trie();

for( i = 0 ; i < n && scanf("%s" , &s[ i ] ) ; ++i )

trie -> insert( trie -> getRoot( ) , s[ i ] , 0 );

int cnt = trie -> countAllWords( trie -> getRoot() );

if( cnt >= n ){

set< string > \_set;

for( i = 0 ; i < n ; ++i ){

for( j = 0 ; j < len ; ++j )

if( s[ i ][ j ] == '1' ) s[ i ][ j ] = '0';

\_set.insert( s[ i ] );

}

if( \_set.size() != n ) puts("NO");

else printf("YES %d\n" , cnt );

}

else puts("NO");

}

puts("YES 0");

return 0;

}

# Bridge Hands

\*\*\*ID: 555

\*\*\*Tipo: Ad hoc , Game - cards

#include <bits/stdc++.h>

using namespace std;

char aux[ 13 ] = { '2' , '3' , '4' , '5' , '6' , '7' , '8' , '9' , 'T' , 'J' , 'Q' , 'K' , 'A' };

bool cmp( string s1 , string s2 ){

if( s1[ 0 ] == s2[ 0 ] ){

for( int i = 12 ; i >= 0 ; --i ){

if( s1[ 1 ] == aux[ i ] ) return 0;

if( s2[ 1 ] == aux[ i ] ) return 1;

}

}

if( s1[ 0 ] == 'H' ) return 0;

if( s2[ 0 ] == 'H' ) return 1;

if( s1[ 0 ] == 'S' ) return 0;

if( s2[ 0 ] == 'S' ) return 1;

if( s1[ 0 ] == 'D' ) return 0;

if( s2[ 0 ] == 'D' ) return 1;

if( s1[ 0 ] == 'C' ) return 0;

if( s2[ 0 ] == 'C' ) return 1;

}

int main(){

char dealer;

char s[ 55 ];

string deck, aux;

int idx;

vector<string> ady[ 4 ];

char d[ 4 ] = {'S' , 'W' , 'N' , 'E' };

while( scanf(" %c" , &dealer ) && dealer !='#' ){

if( dealer == 'N' ) idx = 3;

else if( dealer == 'E' ) idx = 0;

else if( dealer == 'S' ) idx = 1;

else if( dealer == 'W' ) idx = 2;

deck = "";

scanf("%s" , &s );

deck += s;

scanf("%s" , &s );

deck += s;

for( int i = 0 ; i < deck.length() ; i += 2 ){

aux = deck[ i ];

aux += deck[ i + 1 ];

ady[ idx ].push\_back( aux );

idx = ( idx + 1 ) % 4;

}

for( int i = 0 ; i < 4 ; ++i ){

printf("%c:" , d[ i ] );

sort( ady[ i ].begin() , ady[ i ].end() , cmp );

for( int j = 0 ; j < 13 ; ++j ){

printf(" %s" , ady[ i ][ j ].c\_str() );

}

ady[ i ].clear();

printf("\n");

}

}

return 0;

}

# Amazing

\*\*\*ID: 556

\*\*\*Tipo: Simulation

#include <bits/stdc++.h>

using namespace std;

#define MAX 305

#define E 0

#define N 1

#define W 2

#define S 3

//izquierda , arriba , derecha , abajo

int dx[ 4 ] = { 0 , -1 , 0 , 1 };

int dy[ 4 ] = { 1 , 0 , -1 , 0 };

int main(){

int h , w , x , y , dir , nx , ny , lx , ly;

char ady[ MAX ][ MAX ];

int seen[ MAX ][ MAX ];

int ans[ 5 ];

bool out , enter;

while( scanf("%d %d" , &h , &w ) , h | w ){

for( int i = 0 ; i < h ; ++i ) scanf("%s" , ady[ i ] );

x = h - 1; y = 0; dir = E;

memset( seen , 0 , sizeof( seen ) );

out = false;

while( 1 ){

nx = dx[ dir ] + x; ny = dy[ dir ] + y;

//extremos

if( ny == w ){

if( x + dx[ S ] < h && ady[ x + dx[ S ] ] [ y ] == '0' ) dir = S;

else if( x + dx[ N ] < h && ady[ x + dx[ N ] ] [ y ] == '0' ) dir = N;

else dir = W;

out = true;

}

else if( ny == -1 ){

if( x + dx[ N ] >= 0 && ady[ x + dx[ N ] ] [ y ] == '0' ) dir = N;

else if( x + dx[ S ] >= 0 && ady[ x + dx[ S ] ] [ y ] == '0' ) dir = S;

else dir = E;

out = true; }

else if( nx == -1 ){

if( y + dy[ E ] < w && ady[ x ][ y + dy[ E ] ] =='0' ) dir = E;

else if( y + dy[ W ] < w && ady[ x ][ y + dy[ W ] ] =='0' ) dir = W;

else dir = S;

out = true;

}

else if( nx == h ){

if( y + dy[ W ] >= 0 && ady[ x ][ y + dy[ W ] ] =='0' ) dir = W;

else if( y + dy[ E ] < w && ady[ x ][ y + dy[ E ] ] == '0' ) dir = E;

else dir = N;

out = true;

}

//cuando choco a pared

else if( ady[ nx ][ ny ] == '1' ){

out = false;

if( dir == W ){

//caso

//000

//001

//100

if( x + dx[ N ] >= 0 && ady[ x + dx[ N ] ][ y ] == '0') dir = N;

else dir = S;

}else if( dir == S ){ // si choco abajo con pared tengo 2 opciones

if( y + dy[ W ] >= 0 && ady[ x ][ y + dy[ W ] ] == '0') dir = W;

else dir = E;

}

else if( dir == N ){

if( y + dy[ E ] < w && ady[ x ][ y + dy[ E ] ] == '0' ) dir = E;

else dir = W;

}

else if( dir == E ){

if( x + dx[ S ] < h && ady[ x + dx[ S ] ][ y ] == '0') dir = S;

else dir = N;

}

}

//si tengo espacio pa reccorer en otra direccion

else if( x + dx[ N ] != lx && x + dx[ N ] >= 0 && ady[ x + dx[ N ] ][ y ] == '0' && dir == W && !out ){ dir = N;}

else if( y + dy[ W ] != ly && y + dy[ W ] >= 0 && ady[ x ] [ y + dy[ W ] ] == '0' && dir == S && !out ) dir = W; //si bajo y hay espacio en el oeste

else if( y + dy[ E ] != ly && y + dy[ E ] < w && ady[ x ][ y+ dy[ E ] ] =='0' && dir == N && !out )dir = E;

else if( x + dx[ S ] != lx && x + dx[ S ] < h && ady[ x + dx[ S ] ][ y ] == '0' && dir == E && !out ) dir = S;

//11Y

//1 Y tengo q ir al oeste y no abajo

//11

else{

lx = x;

ly = y;

x = nx;

y = ny;

seen[ x ][ y ]++;

out = false;

if( x == h - 1 && y == 0 )break;

}

}

memset( ans , 0 , sizeof( ans ) );

for( int i = 0 ; i < h ; ++i ){

for( int j = 0 ; j < w ; ++j ){

if( ady[ i ][ j ] == '0')ans[ seen[ i ][ j ] ]++;

}

}

printf("%3d%3d%3d%3d%3d\n" , ans[ 0 ] , ans[ 1 ] , ans[ 2 ] , ans[ 3 ] , ans[ 4 ] );

}

return 0;

}

# Wormholes

ID: 558

\*\*\*Tipo: Graph Theory, Bellmand Ford, negative cycle

#include <bits/stdc++.h>

#include <limits.h>

#define MAX 1005

struct edge{

int u, v, w;

};

edge \*e; //e = Arreglo de todas las aristas

long long d[ MAX ]; //Distancias

int n; //Cantidad de nodos

int m; //Cantidad de aristas

//retorna true si no hay ciclos negativos y false en caso q exista ciclo negativo

bool bellman(int &s){

for ( int i = 0 ; i < n ; ++i ) d[ i ] = INT\_MAX;

d[ s ] = 0LL;

// relajamos cada arista del grafo tantas veces como número de nodos -1 haya en el grafo

for ( int i = 0 ; i < n - 1 ; ++i ){

bool cambio = false;

for (int j = 0; j < m ; ++j){

int u = e[ j ].u, v = e[ j ].v;

long long w = e[ j ].w;

if ( d[ u ] + w < d[ v ] ){

d[ v ] = d[ u ] + w;

cambio = true;

}

}

if ( !cambio ) break;

}

// comprobamos si hay ciclos negativo

for (int j = 0 ; j < m ; ++j ){

int u = e[ j ].u, v = e[ j ].v;

long long w = e[ j ].w;

if ( d[ u ] + w < d[ v ] ) return false;

}

return true;

}

int main(){

int t;

scanf("%d", &t);

while( t-- ){

scanf("%d %d", &n ,&m);

e = new edge[ n ];

for ( int i = 0 ; i < m ;++i ){

scanf("%d %d %d",&e[ i ].u , &e[ i ].v , &e[ i ].w);

}

int inicio = 0;

if( bellman( inicio ) )printf("not possible\n");

else printf("possible\n");

}

return 0;

}

# Dividing Coins

\*\*\*ID: 562

\*\*\*Tipo: DP, Sumsets

#include <bits/stdc++.h>

using namespace std;

int main(){

int t, n;

scanf("%d" , &t );

int a[ 105 ], sum, dp[ 25005 ] , m;

while( t-- ){

sum = 0;

scanf("%d" , &n );

for( int i = 0 ; i < n ; ++i ){

scanf("%d" , &a[ i ] );

sum += a[ i ];

}

m = sum/2;

memset( dp , 0 , sizeof( dp ) );

dp[ 0 ] = 1;

for( int i = 0 ; i < n ; ++i ){

for( int j = m ; j >= a[ i ] ; --j ){

if( !dp[ j ] ){

dp[ j ] = dp[ j - a[ i ] ] ;

}

}

}

///basta encontrar el maximo valor para una mitad para saber

///el resto simplemente a la suma total le restamos el valor

///de la maxima mitad

int i = m;

for( ; i >= 1 ; --i ){

if( dp[ i ] ){

break;

}

}

printf("%d\n" , ( sum - i \* 2 ) );

}

return 0;

}

# Crimewave

\*\*\*ID: 563

\*\*\*Tipo: Graph Theory, Max Flow

#include <bits/stdc++.h>

using namespace std;

struct Edge{

int u , v , c , r;

Edge( int uu , int vv ,int cc ,int rr): u( uu ) , v( vv ) , c( cc ) , r( rr ){}

};

vector<Edge> E;

vector<vector<int> > ady;

int maxflow( int source , int sink ){

for( int f = 0; ;){

///vector de enlaces previos para ver la ruta recorrida del argumenting path

///tiene indices de aristas

vector<int> prev( ady.size() , -1 );

queue<int> Q;

Q.push( source );

while( !Q.empty() ){

int u = Q.front(); Q.pop();

for( int i = 0 ; i < ady[ u ].size(); ++i ){

///indice de arista actual

int e = ady[ u ][ i ];

int v = E[ e ].v;

///si no puede pasar o si ya fue visitado

if( E[ e ].c == 0 || prev[ v ] >= 0 )continue;

Q.push( v );

prev[ v ] = e;

}

}

///si ya no hay argumenting paths

if( prev[ sink ] < 0 )return f;

///hallamos minima capacidad de argumentin path

int c = 0x7fffffff;

for( int x = sink ; x != source ; x = E[ prev[ x ] ].u ){

c = min( c , E[ prev[ x ] ].c );//c <?= E[ prev[ x ] ].c;

}

///para nodos de ida 1->2 restamos capacidad minima

///para nodos de vuevla 2<-1 sumamos capacidad minima

for( int x = sink ; x != source ; x = E[ prev[ x ] ].u ){

E[ prev[ x ] ].c -= c;

E[ E[ prev[ x ] ].r ].c += c;

}

f += c;

}

}

void add( int u , int v , int c , bool directed = true ){

Edge e( u , v , c , E.size() + 1 );

ady[ u ].push\_back( E.size() );

E.push\_back( e );

Edge e1( v , u , (directed)? 0 : c , E.size() - 1 );

ady[ v ].push\_back( E.size() );

E.push\_back( e1 );

}

int f , c, b;

int in( int u ){ return 2 \* u;}

int out( int u ){ return 2 \* u + 1;}

int getId( int x , int y ){ return x \* c + y;}

int dx[ 4 ] = { 0 , 0 , 1 , -1 };

int dy[ 4 ] = { 1 , -1 , 0 , 0 };

int main(){

int t , x , y, source , sink , V;

scanf("%d" ,&t );

while( t-- ){

scanf("%d %d %d" , &f , &c , &b );

V = ( f \* c + 1 ) \* 2 + 5;

E.clear();

ady = vector< vector<int> >( V , vector<int> () );

source = f \* c; sink = source + 1;

for( int i = 0 ; i < b ; ++i ){

scanf("%d %d" , &x ,&y );

x--; y--;

add( out( getId( x , y ) ) , in( sink ) , 1 );

}

///unimos con los 4 adyacentes

for( int i = 0 ; i < f ; ++i ){

for( int j = 0 ; j < c ; ++j ){

//nodos en el borde

if( i == 0 || i == f - 1 || j == 0 || j == c - 1 ){

add( out( source ) , in( getId( i , j ) ) , 1 );

}

for( int k = 0 ; k < 4 ; ++k ){

int nx = i + dx[ k ];

int ny = j + dy[ k ];

if( nx >= 0 && nx < f && ny >= 0 && ny < c ){

add( out( getId( i , j ) ) , in( getId( nx , ny ) ) , 1 );

}

}

add( in( getId( i , j ) ) , out( getId( i , j ) ) , 1 );

}

}

add( in( source ) , out( source ) , INT\_MAX/2 );

add( in( sink ) , out( sink ) , INT\_MAX/2 );

int f = maxflow( in( source ) , out( sink ) );

if( f == b )printf("possible\n");

else printf("not possible\n");

}

return 0;

}

# Risk

\*\*\*ID: 567

\*\*\*Tipo: Graph theory, bfs, floyd warshal

#include <bits/stdc++.h>

#include<stdio.h>

#include<queue>

using namespace std;

#define MAX 20

#define INF 9999999

int ady[MAX][MAX];

void init(){

for(int i=0;i<MAX;i++){

for(int j=0;j<MAX;j++){

ady[i][j]=INF;

}

ady[i][i]=0;

}

}

void floyd(){

for(int k=0;k<MAX;k++){

for(int i=0;i<MAX;i++){

for(int j=0;j<MAX;j++){

int t=ady[i][k]+ady[k][j];

if(t<ady[i][j]){

ady[i][j]=t;

}

}

}

}

}

int main(){

int nodos,i,j,cont=1,dest,casos,xi,xf;

while(scanf("%d",&nodos)!=EOF){

init();

for(i=0;i<nodos;i++){

scanf("%d",&dest);

ady[0][dest-1]=ady[dest-1][0]=1;

}

for(i=1;i<=18;i++){

scanf("%d",&nodos);

for(j=0;j<nodos;j++){

scanf("%d",&dest);

ady[i][dest-1]=ady[dest-1][i]=1;

}

}

scanf("%d",&casos);

floyd();

printf("Test Set #%d\n",cont);

while(casos--){

scanf("%d %d",&xi,&xf);

printf("%2d to %2d: %d\n",xi,xf,ady[xi-1][xf-1]);

}

cont++;

putchar('\n');

}

return 0;

}

# Just the facts

\*\*\*ID: 568

\*\*\*Tipo: Math, factorials

#include<stdio.h>

int main(){

int N;

unsigned long long fact = 1;

int factorial[10005];

for(int i=1;i<=10000;i++){

fact \*= i;

while( !(fact % 10) )

fact /= 10;

fact %= 1000000;

factorial[i]=fact;

}

while(scanf("%d",&N)!=EOF){

printf("%5d -> %d\n",N,factorial[N]%10);

}

return 0;

}

# Jugs

\*\*\*ID: 571

\*\*\*Tipo: BFS

#include <bits/stdc++.h>

using namespace std;

#define MAX 1005

int n , A , B;

bool seen[ MAX ][ MAX ];

string s[ 7 ] = { "" , "fill A" , "fill B" , "empty A" , "empty B" , "pour A B" , "pour B A" };

struct Estado{

int a , b;

string exp;

Estado( int aa , int bb ,string s ): a( aa ) , b( bb ) , exp( s ){}

};

void bfs(){

queue< Estado > Q;

Q.push( Estado( 0 , 0 , "" ) );

memset( seen , 0 , sizeof( seen ) );

int a , b;

string exp;

while( !Q.empty() ){

Estado act = Q.front(); Q.pop();

if( seen[ act.a ][ act.b ] ) continue;

a = act.a; b = act.b; exp = act.exp;

if( b == n ){

for( int i = 0 ; i < exp.length() ; ++i ){

printf("%s\n" , s[ exp[ i ] - '0' ].c\_str() );

}

printf("success\n");

return;

}

seen[ a ][ b ] = 1;

if( !seen[ A ][ b ] && a == 0 )Q.push( Estado( A , b , exp + "1" ) );

if( !seen[ a ][ B ] && b == 0 )Q.push( Estado( a , B , exp + "2" ) );

if( !seen[ 0 ][ b ] ) Q.push( Estado( 0 , b , exp + "3" ) );

if( !seen[ a ][ 0 ] ) Q.push( Estado( a , 0 , exp + "4" ) );

if( a > 0 ){

int aux = min( B - b , a );

if( !seen[ a - aux ][ b + aux ] ) Q.push( Estado( a - aux , b + aux , exp + "5" ) );

}

if( b > 0 ){

int aux = min( A - a , b );

if( !seen[ a + aux ][ b - aux ] ) Q.push( Estado( a + aux , b - aux , exp + "6" ) );

}

}

return;

}

int main(){

while( scanf("%d %d %d" , &A ,&B , &n ) != EOF ) bfs();

return 0;

}

# Oil Deposits

\*\*\*ID: 572

\*\*\*Tipo: Graph Theory, BFS, Flood Fill

#include <bits/stdc++.h>

using namespace std;

#define MAX 105

char ady[MAX][MAX];

bool visitado[MAX][MAX];

int h,w;

struct Estado{

int x,y;

Estado(int x1,int y1):x(x1),y(y1){}

};

int dx[8]={1,-1,0, 0,1,-1,-1,1};

int dy[8]={0, 0,1,-1,1, 1,-1,-1};

void bfs(int x, int y, char c){

Estado inicial(x,y);

queue<Estado> Q;

Q.push(inicial);

while(!Q.empty()){

Estado actual=Q.front();Q.pop();

if(visitado[actual.x][actual.y])continue;

visitado[actual.x][actual.y]=true;

for(int i=0;i<8;i++){

int nx=dx[i]+actual.x;

int ny=dy[i]+actual.y;

if(nx>=0 && nx<h && ny>=0 && ny<w && ady[nx][ny]==c && !visitado[nx][ny]){

Estado vecino(nx,ny);

Q.push(vecino);

}

}

}

return ;

}

int main(){

int resp;

string s;

while(scanf("%d %d",&h,&w) && (h!=0 && w!=0)){

for(int i=0;i<h;i++){

for(int j=0;j<w;j++){

ady[i][j]=' ';

visitado[i][j]=false;

}

}

for(int i=0;i<h;i++){

cin>>s;

for(int j=0;j<w;j++){

ady[i][j]=s[j];

}

}

resp=0;

for(int i=0;i<h;i++){

for(int j=0;j<w;j++){

if(!visitado[i][j] && ady[i][j]=='@'){

bfs(i,j,'@');

resp++;

}

}

}

printf("%d\n",resp);

}

return 0;

}

# Sum It Up

\*\*\*ID: 574

\*\*\*Tipo: Bitwise, Sumsets

#include <bits/stdc++.h>

using namespace std;

bool f( vector<int> v1, vector<int> v2 ){

int l = min( v1.size() , v2.size() );

for( int i = 0 ; i < l ; ++i ){

if( v1[ i ] == v2[ i ])continue;

else return v1[ i ] > v2[ i ];

}

}

int main(){

int t, n, sum;

while( scanf("%d %d", &t, &n ) && t|n ){

int a[ n ];

for( int i = 0 ; i < n ; ++i ) scanf("%d", &a[ i ]);

set< vector<int> > v;

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

sum = 0;

vector<int> vv;

for( int j = 0 ; j < n ;++j ){

if( i & 1<<j ){

sum+=a[ j ];

vv.push\_back( a[ j ]);

}

}

if( sum == t ){

sort( vv.rbegin() , vv.rend());

v.insert( vv );

}

}

set< vector<int> > ::iterator it;

printf("Sums of %d:\n", t );

if( v.size() == 0 )printf("NONE\n");

else{

vector< vector<int> > sumas;

for( it = v.begin() ; it != v.end(); ++it ){

sumas.push\_back( \*it);

}

sort( sumas.begin() , sumas.end() , f );

for( int i = 0 ; i < sumas.size(); ++i ){

printf("%d", sumas[ i ][ 0 ]);

for( int j = 1 ; j < sumas[ i ].size(); ++j ){

printf("+%d", sumas[ i ][ j ]);

}

printf("\n");

}

}

}

return 0;

}

# Skew Binary

\*\*\*ID: 575

\*\*\*Tipo: Ad Hoc

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

int main(){

string s;

int i;

long long e,resp;

while(cin>>s){

if(s[0]=='0')break;

e=1;

resp=0;

for(i=s.length()-1;i>=0;i--){

resp+=(s[i]-'0')\*((1<<e) - 1);

e++;

}

cout<<resp<<endl;

}

return 0;

}

# Haiku Review

\*\*\*ID: 576

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

bool isVowel(char a){if(a=='a' || a=='e' || a=='i' || a=='o' || a=='u' || a=='y'){return true;}return false;}

int main(){

char s[205],ant,ef[6]={'e','/','o','/','i','\0'};

int l,i,cont,nexts;

bool vowel;

while(gets(s)){

if(strcmp(s,ef)==0)break;

l=strlen(s);

cont=0;nexts=0;

ant='z';

for(i=0;i<l;i++){

if(i+1==l)nexts++;

vowel=isVowel(s[i]);

if(s[i]=='/'){

nexts++;

if(nexts==1){if(cont!=5) {printf("1\n");break;}}

if(nexts==2){if(cont!=7) {printf("2\n");break;}}

cont=0;

ant=s[i];

continue;

}

if(!vowel){ant=s[i];continue;}

if(vowel && isVowel(ant)){ant=s[i];}

else if(vowel){cont++;ant=s[i];}

}

if(nexts==3){

if(cont!=5){printf("3\n");}

else printf("Y\n");

}

}

return 0;

}

# ClockHands

\*\*\*ID: 579

\*\*\*Tipo: Ad hoc Time

//http://en.wikipedia.org/wiki/Clock\_angle\_problem

#include <bits/stdc++.h>

double abs( double x ){

if( x < 0 )x = -x;

return x;

}

//angulo de la hora partiendo desde 12

double angleHour( int h , int m ){

return ( h \* 60 + m )/2.0;

}

//angulo de la mano de minuto partiendo desde 12

double angleMinute( int m ){

return m \* 6;

}

//angulo entre dos manos desde 12 en sentido horario

double angleHands( int h , int m ){

return abs( ( 60.0 \* h - 11.0 \* m )/2.0 );

}

double min( double a , double b){

if( a < b )return a;

return b;

}

int main(){

int h ,m;

double ang;

while( scanf("%d:%d" , &h , &m ) , h ){

ang = angleHands( h , m );

printf( "%.3lf\n" , min( 360 - ang , ang ) );

}

return 0;

}

# Prime Factors

\*\*\*ID: 583

\*\*\*Tipo: Number Theory

#include <bits/stdc++.h>

using namespace std;

vector<int> factorization(int n){

vector<int> factores;

int i;

for(i=2;i<=n/i;i++){

while(n%i==0){

factores.push\_back(i);

n=n/i;

}

}

if(n>1){factores.push\_back(n);}

return factores;

}

int main(){

int n,aux,i;

vector<int> factores;

while(cin>>n){

if(n<INT\_MIN || n>INT\_MAX || n==1 || n==-1 || n==0){break;}

cout<<n<<" = ";

if(n<0){

cout<<"-1 x ";

aux=n\*(-1);

}

else{aux=n;}

factores=factorization(aux);

for(i=0;i<factores.size();i++){

cout<<factores.at(i);

if(i!=factores.size()-1){

cout<<" x ";

}

}

cout<<endl;

}

return 0;

}

# Bowling

\*\*\*ID: 584

\*\*\*Tipo: Ad hoc simulation

#include <bits/stdc++.h>

char rolls[ 105 ];

int n;

void solve(){

int scores[ 105 ] = { 0 };

int prev = 0;

for( int i = 0 , j = 0 ; i < n ; ++i ){

if( '0' <= rolls[ i ] && rolls[ i ] <= '9' ){

scores[ i ] = rolls[ i ] - '0';

j++;

if( j == 2 ){

j = 0;

}

}

else{

if( j != 0 ){

scores[ i ] = 10 - prev;

j =0;

}else scores[ i ] = 10;

}

prev = scores[ i ];

}

int ans = 0, frames = 0;

for( int i = 0 , j = 0 ; i < n && frames < 10 ; ++i ){

ans += scores[ i ];

if( rolls[ i ] == 'X'){

ans += scores[ i + 1 ] + scores[ i + 2 ];

frames++;

j = 0;

}else if( rolls[ i ] == '/'){

ans += scores[ i + 1 ];

j = 0;

frames++;

}

else{

j++;

if( j == 2 ){

j = 0;

frames++;

}

}

}

printf("%d\n", ans );

}

int main(){

char s[ 105 ];

int l;

while( gets( s ) , s[ 0 ] != 'G'){

l = strlen( s );

n = 0;

for( int i = 0 ; i < l ; ++i ){

if( s[ i ] == ' ')continue;

rolls[ n++ ] = s[ i ];

}

solve();

}

return 0;

}

# Always on the run

\*\*\*ID: 590

\*\*\*Tipo: DP

#include <bits/stdc++.h>

#include <vector>

using namespace std;

#define MAXV 15

#define MAXT 1005

#define MAXD 35

#define INF 1<<30

int V , k;

vector< int > ady[ MAXV ][ MAXV ];

int dp[ MAXT ][ MAXV ];

int solve( int day , int last ){

if( day == k ){

if( last == V - 1 ) return 0;

return INF;

}

if( dp[ day ][ last ] != INF ) return dp[ day ][ last ];

int ans = INF , i , y , w;

for( i = 0 ; i < V ; ++i ){

if( i == last ) continue;

y = day % ady[ last ][ i ].size();

w = ady[ last ][ i ][ y ];

if( w == 0 ) continue;

ans = min( ans , w + solve( day + 1 , i ) );

}

return dp[ day ][ last ] = ans;

}

int main(){

int q , x , i , j , t = 1 , ans;

while( scanf("%d %d" , &V , &k ) , V | k ){

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

if( i == j ) continue;

scanf("%d" , &q );

while( q-- ){

scanf("%d" , &x );

ady[ i ][ j ].push\_back( x );

}

}

}

for( i = 0 ; i < k ; ++i )

for( j = 0 ; j < V ; ++j )

dp[ i ][ j ] = INF;

printf("Scenario #%d\n" , t++ );

ans = solve( 0 , 0 );

if( ans == INF ) puts("No flight possible.");

else printf("The best flight costs %d.\n" , ans );

printf("\n");

for( i = 0 ; i < V ; ++i )

for( j = 0 ; j < V ; ++j )

ady[ i ][ j ].clear();

}

return 0;

}

# Box of bricks

\*\*\*ID: 591

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

int main(){

int n,hi,i,sum,prom,c=1,p,ne;

while(scanf("%d",&n)!=EOF){

if(n==0)break;

int h[n];

sum=0;p=0;ne=0;

for(i=0;i<n;i++){

scanf("%d",&hi);

h[i]=hi;

sum+=hi;

}

prom=sum/n;

for(i=0;i<n;i++){

if(h[i]-prom<0)ne+=(prom-h[i]);

else p+=h[i]-prom;

}

printf("Set #%d\n",c++);

printf("The minimum number of moves is %d.\n\n",std::max(p,ne));

}

return 0;

}

# Bundling Newspapers

\*\*\*ID: 598

\*\*\*Tipo: Backtracking

#include <bits/stdc++.h>

#define MAX 105

char name[ MAX ][ 35 ];

int n;

bool seen[ MAX ];

int ans[ MAX ];

void solve( int len , int last , int sz ){

if( len == sz ){

for( int i = 0 ; i < sz ; ++i ){

if( i ) printf(", ");

printf("%s" , name[ ans[ i ] ] );

}

printf("\n");

return ;

}

for( int i = last ; i < n ; ++i ){

if( seen[ i ] )continue;

seen[ i ] = 1;

ans[ len ] = i;

solve( len + 1 , i , sz );

seen[ i ] = 0;

}

}

int main(){

int t , a , b , l ;

char s[ MAX ] , \*p;

for ( scanf( "%d", &t ); t-- > 0 && scanf(" %[^\n]\n", s ) == 1 ; ) {

if( s[ 0 ] == '\*' ){

a = 1;

b = 0;

}

else if( sscanf( s , "%d %d", &a , &b ) == 1 ) {

b = a;

}

for ( n = 0 ; gets( name[ n ] ) && name[ n ][ 0 ] != '\0'; n++ );

if( b == 0 ) b = n;

for( ; a <= b ; ++a ){

printf("Size %d\n" , a );

memset( seen , 0 , sizeof( seen ) );

solve( 0 , 0 , a );

printf("\n");

}

if( t != 0 ) printf("\n");

}

return 0;

}

# The Forrest for the Trees

\*\*\*ID: 599

\*\*\*Tipo: Graph Theory, Union Find

#include <bits/stdc++.h>

#define MAX 27

int uf[ MAX ];

int Find( int x ){

return ( uf[ x ] == x ) ? x : uf[ x ] = Find( uf[ x ] );

}

void Union( int x , int y ){

uf[ Find( x ) ] = Find( y );

}

int main(){

int t , l, num\_c, acorn;

char s[ 55 ];

bool seen[ 27 ];

scanf("%d" , &t );

gets( s );

while( t-- ){

memset( seen , 0 , sizeof( seen ) );

acorn = 0;

for( int i = 0 ; i < 26 ; ++i )uf[ i ] = i;

while( gets( s ), s[ 0 ] != '\*' ){

Union( s[ 1 ] - 'A' , s[ 3 ] - 'A' );

seen[ s[ 1 ] - 'A' ] = seen[ s[ 3 ] - 'A' ] = 1;

}

gets( s );

l = strlen( s );

num\_c = 0;

for( int i = 0 ; i < l ; ++i ){

if( s[ i ] != ',' && !seen[ s[ i ] - 'A' ] ){

acorn++;

}

}

for( int i = 0 ; i < 26 ; ++i ){

if( seen[ i ] && Find( i ) == i ){

num\_c++;

}

}

printf("There are %d tree(s) and %d acorn(s).\n" , num\_c , acorn );

}

return 0;

}

# The Path

\*\*\*ID: 601

\*\*\*Tipo: Graph Theory, BFS 0/1

#include <bits/stdc++.h>

using namespace std;

#define MAX 81

char ady[ MAX ][ MAX ];

int n;

bool seen[ MAX ][ MAX ];

int dist[ MAX ][ MAX ];

struct Estado{

int x ,y;

Estado( int xx , int yy ): x( xx ) , y( yy ){}

};

int dx[ 4 ] = { 1 , -1 , 0 , 0 };

int dy[ 4 ] = { 0 , 0 , 1 , -1 };

void bfs( int x , int y , char c ){

memset( dist , -1 , sizeof( dist ) );

deque<Estado> Q;

Q.push\_front( Estado( x , y ) );

dist[ x ][ y ] = 0;

while( !Q.empty() ){

Estado act =Q.front(); Q.pop\_front();

for( int i = 0 ; i < 4 ; ++i ){

int nx = act.x + dx[ i ];

int ny = act.y + dy[ i ];

if( nx >= 0 && nx < n && ny >= 0 && ny < n && dist[ nx ][ ny ] == -1 ){

if( ady[ nx ][ ny ] == 'U' ){

Q.push\_back( Estado( nx , ny ) );

dist[ nx ][ ny ] = dist[ act.x ][ act.y ] + 1;

}

else if( ady[ nx ][ ny ] == c){

Q.push\_front( Estado( nx , ny ) );

dist[ nx ][ ny ] = dist[ act.x ][ act.y ];

}

}

}

}

}

int main(){

int resp;

bool winner;

while( scanf("%d" , &n ) && n ){

for( int i = 0 ; i < n ; ++i ){

scanf("%s" , &ady[ i ] );

}

if( n == 1 && strcmp( ady[ 0 ] , "U") == 0 ){

printf("White can win in one move.\n");

continue;

}

winner = false;

for( int i = 0 ; i < n ; ++i ){

if( ady[ i ][ 0 ] != 'W' )continue;

bfs( i , 0 , 'W' );

///revisamos si se llego al final

for( int j = 0 ; j < n ; ++j ){

if( dist[ j ][ n - 1 ] == 0 ){

printf("White has a winning path.\n");

winner = true;

break;

}

}

if( winner )break;

for( int j = 0 ; j < n ; ++j ){

if( dist[ j ][ n - 1 ] == 1 ){

printf("White can win in one move.\n");

winner = true;

break;

}

}

if( winner )break;

}

if( winner ) continue;

for( int i = 0 ; i < n ; ++i ){

if( ady[ 0 ][ i ] != 'B' )continue;

bfs( 0 , i , 'B' );

///revisamos si se llego al final

for( int j = 0 ; j < n ; ++j ){

if( dist[ n - 1 ][ j ] == 0 ){

printf("Black has a winning path.\n");

winner = true;

break;

}

}

if( winner )break;

for( int j = 0 ; j < n ; ++j ){

if( dist[ n - 1 ][ j ] == 1 ){

printf("Black can win in one move.\n");

winner = true;

break;

}

}

if( winner )break;

}

if( !winner ) printf("There is no winning path.\n");

}

}

# Stacking Boxes

\*\*\*ID: 103

\*\*\*Tipo: Sorting, LIS

#include <bits/stdc++.h>

using namespace std;

#define mp make\_pair

#define pb push\_back

#define pvi pair< vector< int > , int >

vector< pvi > v;

vector< int > :: iterator it;

int k;

int dp[ 35 ] , prev[ 35 ];

bool entra( int x , int y ){

for( int i = 0 ; i < k ; ++i )

if( v[ x ].first[ i ] >= v[ y ].first[ i ] ) return false;

return true;

}

void print( int idx , int len ){

if( prev[ idx ] == -1 || len <= 0 ){ printf("%d" , v[ idx ].second );return; }

print( prev[ idx ] , len - 1 );

printf(" %d" , v[ idx ].second );

}

int main(){

int n ,x;

while( scanf("%d %d" , &n , &k ) != EOF ){

v.clear();

vector< int > aux;

for( int i = 0 ; i < n ; ++i ){

aux.clear();

for( int j = 0 ; j < k ; ++j ){

scanf("%d" , &x );

aux.push\_back( x );

}

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

v.pb( mp( aux , i + 1 ) );

dp[ i ] = 1;

prev[ i ] = -1;

}

stable\_sort( v.begin() , v.end() );

for( int i = 0 ; i < n ; ++i ){

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

if( entra( i , j ) ){ //si todos los elementos de la caja i entran en la caja j

if( dp[ j ] < dp[ i ] + 1 ){

dp[ j ] = dp[ i ] + 1;

prev[ j ] = i;

}

}

}

}

int maxi = 0, idx = 0;

for( int i = 0 ; i < n ; ++i ){

if( maxi <= dp[ i ] ){

maxi = dp[ i ];

idx = i;

}

}

printf("%d\n" , maxi );

print( idx , maxi );

printf("\n");

}

return 0;

}

# Arbitrage

\*\*\*ID: 104

\*\*\*Tipo: Floyd Warshall

#include <bits/stdc++.h>

#include <string.h>

#define MAX 25

#define INF 1<<30

int V , path[ MAX ][ MAX ][ MAX ];

double ady[ MAX ][ MAX ][ MAX ];

void init2(){

int i , j;

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

path[ i ][ j ][ 1 ] = i;

}

}

}

//FLOYD 3 dimensiones

//i: vertice origen

//j: vertice destino

//s: numero de aristas entre origen y destino

//ady[ i ][ j ][ s ] = maximo o minimo costo entre i y j usando s pasos.

void floydSteps(){

int i , j , k , s;

for( s = 2 ; s <= V ; ++s ){

for( k = 0 ; k < V ; ++k ){

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

if( ady[ i ][ k ][ s - 1 ] \* ady[ k ][ j ][ 1 ] > ady[ i ][ j ][ s ] ){

ady[ i ][ j ][ s ] = ady[ i ][ k ][ s - 1 ] \* ady[ k ][ j ][ 1 ];

path[ i ][ j ][ s ] = k;

}

}

}

}

}

}

void print( int x , int y , int s ){

if( s == 0 ) printf("%d" , x + 1 );

else{

print( x , path[ x ][ y ][ s ] , s - 1 );

printf(" %d" , y + 1 );

}

return;

}

int main(){

int i , j , s;

double w;

while( scanf("%d" , &V ) != EOF ){

memset( ady , 0 , sizeof( ady ) );

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

if( i == j ) { ady[ i ][ j ][ 1 ] = 1.0;}

else{

scanf("%lf" , &w );

ady[ i ][ j ][ 1 ] = w;

}

}

}

init2();

floydSteps();

for( s = 0 ; s <= V ; ++s ){

for( i = 0 ; i < V ; ++i ){

if( ady[ i ][ i ][ s ] > 1.01 ){

print( i , i , s );

break;

}

}

if( i != V ) break;

}

if( s == V + 1 ) printf("no arbitrage sequence exists");

printf("\n");

}

return 0;

}

# Unidirectional TSP

\*\*\*ID: 116

\*\*\*Tipo: DP

#include <bits/stdc++.h>

using namespace std;

int a[ 21 ][ 105 ], dp[ 21 ][ 105 ];

struct Estado{

int x, y;

Estado( int xx , int yy ):x( xx ) , y( yy ){}

Estado(){}

}prev[ 21 ][ 105 ];

int main(){

int h , w, idx\_d , idx\_u;

while( scanf("%d %d" , &h, &w ) == 2 ){

for( int i = 1 ; i <= h ; ++i ){

for( int j = 1 ; j <= w ; ++j ){

scanf( "%d" , &a[ i ][ j ] );

dp[ i ][ j ] = 1<<30;

}

}

for( int k = 0 ; k <= h ; ++k ) {

dp[ k ][ w ] = a[ k ][ w ];

prev[ k ][ 0 ] = Estado( -1 , - 1 );

}

for( int j = w - 1 ; j >= 1; --j ){

for( int i = 1 ; i <= h ; ++i ){

idx\_d = i + 1;

if( i == h ){

idx\_d = 1;

}

idx\_u = i - 1;

if( i == 1 ){

idx\_u = h;

}

dp[ i ][ j ] = dp[ idx\_d ][ j + 1 ] + a[ i ][ j ];

prev[ i ][ j ] = Estado( idx\_d , j + 1 );

if( dp[ i ][ j ] > dp[ i ][ j + 1 ] + a[ i ][ j ] ){

dp[ i ][ j ] = dp[ i ][ j + 1 ] + a[ i ][ j ];

prev[ i ][ j ] = Estado( i , j + 1 );

}

else if( dp[ i ][ j ] == dp[ i ][ j + 1 ] + a[ i ][ j ] ){

if( prev[ i ][ j ].x > i )

prev[ i ][ j ] = Estado( i , j + 1 );

}

if( dp[ i ][ j ] > dp[ idx\_u ][ j + 1 ] + a[ i ][ j ] ){

dp[ i ][ j ] = dp[ idx\_u ][ j + 1 ] + a[ i ][ j ];

prev[ i ][ j ] = Estado( idx\_u , j + 1 );

}

else if( dp[ i ][ j ] == dp[ idx\_u ][ j + 1 ] + a[ i ][ j ] ){

if( prev[ i ][ j ].x > idx\_u )

prev[ i ][ j ] = Estado( idx\_u , j + 1 );

}

}

}

int min = 1<<30, x = 0 ;

for( int i = 1 ; i <= h ; ++i ){

if( min > dp[ i ][ 1 ] ){

x = i;

min = dp[ i ][ 1 ];

}

}

int resp[ 105 ], len = 0;

for( int i = 0 ; i < w ; ++i ){

resp[ i ] = x;

x = prev[ x ][ i + 1 ].x;

}

printf("%d" , resp[ 0 ] );

for( int i = 1 ; i < w ; ++i ) printf(" %d" , resp[ i ] );

printf("\n%d\n" , min );

}

return 0;

}

# The Postal Worker Rings Once

\*\*\*ID: 117

\*\*\*Tipo: Chinese Postman Problem - Euler Tour

#include <bits/stdc++.h>

using namespace std;

#define MAX 105

#define MAXV 27

#define INF 1<<30

struct Edge{

int v , w;

int id;

Edge( int vv , int ww ): v( vv ) , w(ww){}

Edge( int vv , int ww , int i ): v( vv ) , w(ww) , id( i ){}

Edge(){}

};

char s[ MAX ];

int degree[ MAXV ];

vector< Edge > ady[ MAXV ];

int dist[ MAX ] , prev[ MAX ], seen[ MAX ];

void spfa( int source ){

int u , v , w , i;

for( i = 0 ; i < MAXV ; ++i ){

seen[ i ] = 0;

dist[ i ] = INF;

}

queue< int > Q;

dist[ source ] = 0;

Q.push( source );

while( !Q.empty() ){

u = Q.front(); Q.pop();

seen[ u ] = 0;

for( i = 0 ; i < ady[ u ].size() ; ++i ){

v = ady[ u ][ i ].v;

w = ady[ u ][ i ].w;

if( dist[ v ] > dist[ u ] + w ){

dist[ v ] = dist[ u ] + w;

prev[ v ] = u;

if( !seen[ v ] ){

seen[ v ] = 1;

Q.push( v );

}

}

}

}

}

int main(){

long long ans;

int len , source , sink , i;

while( scanf("%s" , &s ) == 1 ){

ans = 0;

len = strlen( s );

if( s[ 0 ] == s[ len - 1 ] ) continue;

memset( degree , 0 , sizeof( degree ) ) ;

ady[ s[ 0 ] - 'a' ].push\_back( Edge( s[ len - 1 ] - 'a' , len ) );

ady[ s[ len - 1 ] - 'a' ].push\_back( Edge( s[ 0 ] - 'a' , len ) );

degree[ s[ 0 ] - 'a' ]++;

degree[ s[ len - 1 ] - 'a' ]++;

ans += len;

while( scanf("%s" , &s ) == 1 ){

len = strlen( s );

if( s[ 0 ] == s[ len - 1 ] ) break;

ady[ s[ 0 ] - 'a' ].push\_back( Edge( s[ len - 1 ] - 'a' , len ) );

ady[ s[ len - 1 ] - 'a' ].push\_back( Edge( s[ 0 ] - 'a' , len ) );

degree[ s[ 0 ] - 'a' ]++;

degree[ s[ len - 1 ] - 'a' ]++;

ans += len;

}

source = sink = -1;

for( i = 0 ; i < MAXV ; ++i ){

if( degree[ i ] & 1 ){

if( source == -1 ) source = i;

else sink = i;

}

}

spfa( source );

ans += dist[ sink ];

printf("%lld\n" , ans );

for( i = 0 ; i < MAXV ; ++i ) ady[ i ].clear();

}

return 0;

}

# Mutant Flatworld Explorers

\*\*\*ID: 118

\*\*\*Tipo: Simulation

#include <bits/stdc++.h>

#define MAX 55

int h , w;

int ady[ MAX ][ MAX ];

char s[ 105 ];

#define N 0

#define W 1

#define S 2

#define E 3

char d[4] = { 'N' , 'W' , 'S' , 'E' };

int dx[4] = { 0 , -1 , 0 , 1 };

int dy[4] = { 1 , 0 , -1 , 0 };

void simulate( int x , int y , char dir ){

for( int i = 0 ; s[ i ] ; ++i ){

if( s[ i ] == 'F' ){

int nx = dx[ dir ] + x, ny = dy[ dir ] + y;

if( nx >= 0 && nx <= h && ny >= 0 && ny <= w ){

x = nx;

y = ny;

}

else{

if( ady[ x ][ y ] ){

continue;

}

else{

printf("%d %d %c LOST\n" , x , y , d[ dir ] );

ady[ x ][ y ] = 1;

return;

}

}

}

else if( s[ i ] == 'L' ){

dir = (dir + 1) % 4;

}

else {

dir = ( 4 + dir - 1 ) % 4;

}

}

printf("%d %d %c\n" , x , y , d[ dir ] );

}

int main(){

int x , y;

char c;

scanf("%d %d" , &h , &w );

memset( ady , 0 , sizeof( ady ) );

while( scanf("%d %d %c" , &x , &y , &c ) != EOF ){

gets( s );

gets( s );

simulate( x , y , ( c=='N')?0:( c == 'S')?2:( c == 'E')?3:1 );

}

return 0;

}

# Stacks of Flapjacks

\*\*\*ID: 120

\*\*\*Tipo: Stacks, Greedy

#include <bits/stdc++.h>

using namespace std;

#define MAX 205

#define MAXN 105

char line[ MAX ];

int a[ MAXN ], b[ MAXN ] , l;

bool cmp( int a , int b ){ return a > b;}

int getIdx( int x ){ return l - x; }

void print(){

for( int i = 0 ; i < l ; ++i ) printf("%d " , a[ i ] ) ;

printf("\n");

}

void flip( int idx ){

int left = 0 , right = idx + 1;

while( left < right ){

int tmp = a[ left ];

a[ left ] = a[ right - 1 ];

a[ right - 1 ] = tmp;

left++; right--;

}

//print();

}

void solve(){

int i , len = l , j;

sort( b , b + l , cmp );

for( i = 0 ; i < l ; ++i ){

for( j = 0 ; j < len ; ++j ){

if( b[ i ] == a[ j ] ) break;

}

if( j != len - 1 ){

if( j == 0 ){

flip( len - 1 );

printf("%d " , l - len + 1 );

}

else {

flip( j );

printf("%d " , l - j );

flip( len - 1 );

printf("%d " , l - len + 1 );

}

}

len--;

}

printf("0\n");

}

int main(){

int x;

while( gets( line ) ){

l = 0;

stringstream ss( line );

while( ss>>x ){ a[ l ] = b[ l++ ] = x; }

for( int i = 0 ; i < l ; ++i ){

if( i ) printf(" ");

printf("%d" , a[ i ] );

}

printf("\n");

solve();

}

return 0;

}

# Trees on the level

\*\*\*ID: 122

\*\*\*Tipo: Ad hoc, Trees traversal

#include <bits/stdc++.h>

using namespace std;

#define MAX 80005

int tree[ MAX ];

int values[ MAX ];

bool visited[ MAX ];

bool tree\_real[ MAX ];

int Int( string s ){

stringstream ss(s);

int n;

ss>>n;

return n;

}

int main(){

string s;

int nodos = 0;

memset( tree, 0 , sizeof( tree ) );

tree\_real[ 0 ] = true;

while( getline( cin , s) ){

s+=" ";

stringstream ss( s );

bool b = false;

while( ss>>s ){

if(s == "()"){b = true;break;}

int it = s.find( "," );

int nodo = Int( s.substr( 1 , it ) );

string ruta = s.substr( it + 1 );

ruta = ruta.substr(0, ruta.length() - 1 );

int x = 0;

for( int i = 0 ; i < ruta.length() ; ++i ){

if( ruta[ i ] == 'L' )

x = 2 \* x + 1;

else

x = 2 \* x + 2;

tree\_real[ x ] = true;

}

tree[ x ]++;

values[ x ] = nodo;

visited[ x ] = true;

nodos = std::max( nodos , x );

}

if( b ){

int i = 0;

for( ; i <= nodos ; ++i ){

if( tree\_real[ i ] && !visited[ i ] || visited[ i ] && tree[ i ] != 1 ){

cout<<"not complete"<<endl;

break;

}

}

if( i == nodos + 1){

cout<<values[ 0 ];

for( i = 1 ; i <= nodos ; ++i ){

if( values[ i ] )

cout<<" "<<values[ i ];

}

cout<<endl;

}

nodos = 0;

memset( tree , 0 , sizeof( tree ) );

memset( values, 0 , sizeof( values ) );

memset( visited, 0 , sizeof( visited ) );

memset( tree\_real , 0 , sizeof( tree\_real ) );

tree\_real[ 0 ] = true;

}

}

return 0;

}

# Following Orders

\*\*\*ID: 124

\*\*\*Tipo: Graph Theory, Topological Sort, Backtracking

#include <bits/stdc++.h>

#include <sstream>

using namespace std;

#define MAX 35

int used[ MAX ];

///reglas representadas con entero a < b -> [ 0 ][ 1 ] = true

int contraint[ MAX ][ MAX ];

char let[ MAX ];

int idx;

char output[ MAX + 5 ];

bool check( int len , int in ){

//si x ejemplo tengo a b f g

//contrait a <b b < f

//primero entra "a" y se compara con "b" "f" y "g"

//contradiccion al contrait es q a > b x ello output va en indice j

for( int i = 0 ; i < len ; ++i ){

if( contraint[ in ][ output[ i ] - 'a' ] ) return false;

}

return true;

}

void dfs( int len ){

if( len == idx ){

output[ len ] ='\0';

printf("%s\n", output );

return;

}

for( int i = 0 ; i < idx ; ++i ){

if( !used[ i ] && check( len , let[ i ] - 'a' ) ){

output[ len ] = let[ i ];

used[ i ] = 1;

dfs( len + 1 );

used[ i ] = 0;

}

}

}

int main(){

char line[ MAX + 5 ];

char a ,b;

int l;

bool bb = false;

while( gets( line ) ){

if( bb ) putchar( '\n' );

bb = true;

idx = 0;

for( int i = 0 ; line[ i ] ; ++i ){

if( isalpha( line[ i ] ) ) {

let[ idx ] = line[ i ];

used[ idx++ ] = 0;

}

}

//se ordena para sacar lexicograficamente

sort( let, let + idx );

memset( contraint , 0 , sizeof( contraint ) );

gets( line );

strcat( line, " ");

stringstream ss( line );

while( ss>>a>>b ) contraint[ a - 'a' ][ b - 'a' ] = 1;

dfs( 0 );

}

return 0;

}

# Numbering Paths

\*\*\*ID: 125

\*\*\*Tipo: Graph Theory, Strongly Connected Components, Transitive Closure

#include <bits/stdc++.h>

#include <string.h>

#define MAX 70

int ady[ MAX ][ MAX ][ MAX ] , V , resp[ MAX ][ MAX ][ MAX ];

void CountingPath(){

int s , i , j , k;

for( s = 2 ; s <= 64 ; ++s ){

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

for( k = 0 ; k < V ; ++k )

ady[ i ][ j ][ s ] += ady[ i ][ k ][ s - 1 ] \* ady[ k ][ j ][ 1 ];

resp[ i ][ j ][ s ] = resp[ i ][ j ][ s - 1 ];

//Si existen rutas mayores al numero de vertices entonces hay ciclo

if( s > V && ady[ i ][ j ][ s ] != 0 ){

resp[ i ][ j ][ s ] = -1;

}else{

resp[ i ][ j ][ s ] += ady[ i ][ j ][ s ];

}

}

}

}

for( i = 0 ; i < V ; ++i ){

for( j = 0 ; j < V ; ++j ){

( j == 0 ) ? printf("%d" , resp[ i ][ j ][ 64 ] ) : printf(" %d", resp[ i ][ j ][ 64 ]);

}

printf("\n");

}

}

int max( int a , int b ){ return ( a < b )? b:a ;}

int main(){

int n, q = 0 , u , v , i , j;

while( scanf("%d" , &n ) != EOF ){

V = 0;

memset( ady , 0 , sizeof( ady ) );

memset( resp , 0 , sizeof( resp ) );

for( i = 0 ; i < n && scanf("%d %d" , &u , &v ) == 2 ; ++i ){

ady[ u ][ v ][ 1 ]++;

resp[ u ][ v ][ 1 ]++;

V = max( u , max( v , V ) );

}

V++;

printf("matrix for city %d\n" , q++ );

CountingPath();

}

return 0;

}

# Roman Roulette

\*\*\*ID: 130

\*\*\*Tipo: Simulation, Josephus

#include <bits/stdc++.h>

#include <iostream>

#include <vector>

using namespace std;

#define MAX 105

int main( ){

int n , k , next , kill , k\_next , nn;

while( scanf("%d %d" , &n , &k ) , n|k ){

if( n == 1 ){

printf("1\n");

continue;

}

nn = n;

vector<int> Circle;

for( int i = 0 ; i < n ; ++i )

Circle.push\_back( i );

k--;

for( int i = 0 ; ; ){

kill = ( i + k ) % n;

Circle.erase( Circle.begin() + kill );

n = Circle.size();

k\_next = ( kill + k ) % n;

Circle.insert( Circle.begin() + kill , Circle[ k\_next ] );

if( k\_next >= kill ) k\_next++;

Circle.erase( Circle.begin() + k\_next );

n = Circle.size();

if( n == 1 )break;

if( k\_next >= kill ){

i = ( kill + 1 ) % n;

}

else{

i = kill % n;

}

}

printf("%d\n" , ( ( nn - Circle[ 0 ] ) % nn ) + 1 );

}

return 0;

}

# The Dole Queue

\*\*\*ID: 133

\*\*\*Tipo: Simulation

#include <bits/stdc++.h>

bool circle[ 24 ];

int main(){

int k , n , m , next\_left , next\_right , l , r, q , len;

while( scanf("%d %d %d" , &n , &k , &m ) , k| m | n ){

memset( circle , 0 , sizeof( circle ) );

r = 0; l = n - 1;

k--;

m--;

q = 0;

len = n;

while( len ){

for( int j = k % len ; ; r++ ){

if( r == n ) r = 0;

if( !circle[ r ] && j-- <= 0 )break;

}

for( int j = m % len ; ; l-- ){

if( l == -1 ) l = n - 1;

if( !circle[ l ] && j-- <= 0 )break;

}

circle[ r ] = circle[ l ] = 1;

if( q++ )printf(",");

if( r == l ){

printf("%3d" , r + 1 );

len--;

}

else{

printf("%3d%3d" , r + 1 , l + 1 );

len -= 2;

}

}

printf("\n");

}

return 0;

}

# Bandwidth

\*\*\*ID: 140

\*\*\*Tipo: Backtracking

#include <bits/stdc++.h>

using namespace std;

#define MAX 27

vector<int> ady[ MAX ];

bool seen[ MAX ], visited[ MAX ], connected[ MAX ][ MAX ];

int pos[ MAX ];

int len, mini;

string resp;

void dfs( int x , string s ){

if( s.length() == len ){

for( int i = 0 ; i < s.length() ; ++i ){

pos[ s[ i ] - 'A' ] = i + 1;

}

int maxi = 0;

for( int i = 0 ; i < MAX ; ++i ){

if( visited[ i ] ){

for( int j = 0 ; j < ady[ i ].size() ; ++j ){

if( !connected[ i ][ ady[ i ][ j ] ] ) continue;

int aux = abs( pos[ ady[ i ][ j ] ] -pos[ i ] );

if( maxi < aux ){

maxi = aux;

}

}

}

}

if( mini > maxi ){

resp = s;

mini = maxi;

}

}

seen[ x ] = 1;

for( int i = 0 ; i < ady[ x ].size() ; ++i ){

if( !seen[ ady[ x ][ i ] ] ){

dfs( ady[ x ][ i ] , s + (char)(ady[ x ][ i ]+'A') );

}

}

seen[ x ] = 0;

}

int main(){

char line[ 105 ];

string s;

while( gets( line ), line[ 0 ] != '#' ){

s = "";

memset( visited , 0 , sizeof( visited ) );

memset( connected , 0 , sizeof( connected) ) ;

for( int i = 0 ; line[ i ] ; ++i ){

if( line[ i ] == ';' ){

for( int j = 2 ; j < s.length() ; ++j ){

connected[ s[ 0 ] - 'A' ][ s[ j ] - 'A' ] = 1;

connected[ s[ j ] - 'A' ][ s[ 0 ] - 'A' ] = 1;

visited[ s[ j ] - 'A' ] = 1;

}

visited[ s[ 0 ] - 'A' ] = 1;

s = "";

}

else s+= line[ i ];

}

for( int j = 2 ; j < s.length() ; ++j ){

connected[ s[ 0 ] - 'A' ][ s[ j ] - 'A' ] = 1;

connected[ s[ j ] - 'A' ][ s[ 0 ] - 'A' ] = 1;

visited[ s[ j ] - 'A' ] = 1;

}

visited[ s[ 0 ] - 'A' ] = 1;

for( int i = 0 ; i < MAX ; ++i ){

if( !visited[ i ] ) continue;

for( int j = i + 1 ; j < MAX ; ++j ){

if( !visited[ j ] ) continue;

ady[ i ].push\_back( j );

ady[ j ].push\_back( i );

}

}

len = 0;

string ini;

int ini\_;

for( int i = 0 ; i < MAX ; ++i ){

if( visited[ i ] ){

len++;

}

}

mini = 1000;

for( int i = 0 ; i < MAX ; ++i ){

ini = "";ini += (char)( i + 'A' );

if( visited[ i ]){

memset( seen , 0 , sizeof( seen ) );

dfs( i , ini );

}

}

for( int i = 0 ; i < len ; ++i )

printf("%c " , resp[ i ] );

printf("-> %d\n" , mini );

for( int i = 0 ; i < MAX ; ++i )ady[ i ].clear();

}

return 0;

}

# Dollars

\*\*\*ID: 147

\*\*\*Tipo: DP, Coin Change

#include <bits/stdc++.h>

using namespace std;

#define MAX 6005

int a[ 11 ] = { 1, 2 , 4 , 10 , 20 , 40 , 100 , 200 , 400 , 1000 , 2000 };

long long dp[ MAX ];

void CoinChange(){

dp[ 0 ] = 1;

for( int i = 0 ; i < 11 ; ++i ){

for( int j = a[ i ] ; j < MAX ; ++j ){

dp[ j ] = dp[ j ] + dp[ j - a[ i ] ];

}

}

}

int main(){

double n;

CoinChange();

while( scanf("%lf" , &n ) && n != 0 ){

printf("%6.2lf%17lld\n" , n , dp[ (int)( n \* 20 ) ] );

}

return 0;

}

# Anagram checker

\*\*\*ID: 148

\*\*\*Tipo: Backtracking, strings

#include <bits/stdc++.h>

#include <cstdio>

#include <cstdlib>

#include <ctime>

#include <functional>

#include <numeric>

#include <utility>

#include <deque>

#include <stack>

#include <bitset>

#include <map>

#include <set>

#include <string>

#include <vector>

#include <queue>

#include <limits>

#include <fstream>

#include <list>

#include <sstream>

#include <iostream>

#include <iomanip>

using namespace std;

#define MAX 3015

char dictionary[ MAX ][ 42 ];

int cantidad[ MAX ][ 27 ];

int a[ 27 ];

char paragraph[ 42 ];

char resp[ MAX ][ 42 ];

bool seen[ MAX ];

int len , len\_anagram;

char anagram[ 25 ][ 22 ];

void solve( int word , int lon ){

int i;

for( i = 0 ; i < 26 ; ++i ){

if( a[ i ] )break;

}

if( i == 26 ){

int cnt = 0;

for( int k = 0 ; k < lon ; ++k ){

for( int i = 0 ; i < len\_anagram ; ++i ){

if( !strcmp( anagram[ i ] , resp[ k ] ) ){

cnt++;

}

}

}

//si mi respuesta contiene las letras en el mismo orden que la original no va

if( cnt == len\_anagram )return;

printf("%s=" , paragraph );

for( int j = 0 ; j < lon ; ++j ){

printf(" %s" , resp[ j ] );

}

printf("\n");

return;

}

if( word >= len ||lon >= len ){

return;

}

for( int i = word ; i < len ; ++i ){

int j;

for( j = 0 ; j < 26 ; ++j ){

if( a[ j ] >= cantidad[ i ][ j ] )continue;

else break;

}

//si tiene todos los caracteres

if( j == 26 ){

for( j = 0 ; j < 26 ; ++j ){

a[ j ] -= cantidad[ i ][ j ];

}

seen[ i ] = true;

strcpy( resp[ lon ] , dictionary[ i ] );

solve( i + 1 , lon + 1 );

//backtrack

seen[ i ] = false;

for( j = 0 ; j < 26 ; ++j ){

a[ j ] += cantidad[ i ][ j ];

}

}

}

}

int main(){

int lp = 0;

len = 0;

memset( cantidad , 0 , sizeof( cantidad ) );

while( gets( dictionary[ len ] ) && dictionary[ len ][ 0 ] != '#'){

int l = strlen( dictionary[ len ] );

for( int i = 0 ; i < l ; ++i ){

if( 'A' <= dictionary[ len ][ i ] && dictionary[ len ][ i ] <= 'Z' )

cantidad[ len ][ dictionary[ len ][ i ] - 'A' ]++;

}

len++;

}

int aux;

while( gets( paragraph ) , strcmp( paragraph , "#") ){

lp = strlen( paragraph );

len\_anagram = 0;

aux = 0;

memset( seen , 0 , sizeof( seen ) );

bool enter = false;

memset( a , 0 , sizeof( a ) );

paragraph[ lp ] = ' ';

paragraph[ lp + 1 ] = '\0';

for( int i = 0 ; i <= lp ; ++i ){

if( 'A' <= paragraph[ i ] && paragraph[ i ] <= 'Z' ){

anagram[ len\_anagram ][ aux++ ] = paragraph[ i ];

anagram[ len\_anagram][aux] = '\0';

a[ paragraph[ i ] - 'A' ]++;

enter = true;

}

else{

if( enter )len\_anagram++;

enter = false;

aux = 0;

}

}

solve( 0 , 0 );

}

return 0;

}

# Counterfeit Dollar

\*\*\*ID: 608

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

int main(){

int t, len;

bool even[ 27 ];

char l[ 3 ][ 27 ] , r[ 3 ][ 27 ] , go[ 3 ][ 5 ];

scanf("%d" , &t );

while( t-- ){

memset( even , 0 , sizeof( even ) );

for( int i = 0 ; i < 3 ; ++i ){

scanf("%s %s %s" , &l[ i ] , &r[ i ] , &go[ i ] );

if( go[ i ][ 0 ] =='e' )

{

len = strlen( l[ i ] );

for( int j = 0 ; j < len ; ++j ){

even[ l[ i ][ j ] - 'A' ] = even[ r[ i ][ j ] - 'A' ] = 1;

}

}

}

//asumo q letter es el incorrecto, verifico si cumple con las condiciones

for( int letter = 'A' ; letter <= 'L' ; ++letter ){

if( even[ letter - 'A' ] ) continue;

int left = -1; //-1 nada 1 left 2 right

int up = -1; //-1 nada, 1 up, 2 down

int cnt = 0; //apariciones de caracter, si tengo una comparacion de up o down con otro caracter

//aparte del q supongo entonces no es.

for( int i = 0 ; i < 3 ; ++i ){

if( go[ i ][ 0 ] =='e' ){ cnt++; continue;}

len = strlen( l[ i ] );

left = -1;

//si caracter esta al lado izquierdo

for( int j = 0 ; j < len ; ++j ){

if( l[ i ][ j ] == letter ){

left = 1;

break;

}

else if( r[ i ][ j ] == letter ){ //si esta al lado derecho

left = 2;

break;

}

}

if( left == -1 )continue;

if( up != -1 ){

//si el q supongo es mas pesado y me contradice diciendo q esta en los livianos, entonces noes

if( left == 1 ){

if( up == 1 && go[ i ][ 0 ] == 'u' )break;

else if( up == 2 && go[ i ][ 0 ] == 'd')break;

}

else if( left == 2 ){

if( up == 1 && go[ i ][ 0 ] == 'd' )break;

else if( up == 2 && go[ i ][ 0 ] == 'u')break;

}

cnt++;

continue;

}

cnt++;

//se ejecutara una vez para ver si es pesado o liviano

if( left == 2){

if( go[ i ][ 0 ] == 'u')up = 1;

else up = 2;

}

else if( left == 1 ){

if( go[ i ][ 0 ] == 'u')up = 2;

else up = 1;

}

}

if( cnt == 3 ){

printf("%c is the counterfeit coin and it is " , letter );

if( up == 1 )printf("light.\n");

else printf("heavy.\n");

break;

}

}

}

return 0;

}

# Street Directions

\*\*\*ID: 610

\*\*\*Tipo: Bridge, Back Edges

#include <bits/stdc++.h>

#include <vector>

using namespace std;

#define MAX 1005

int V, dfsNum[ MAX ] , low[ MAX ], num,seen[ MAX ];

vector< int > ady[ MAX ];

void dfs( int x , int parent ){

dfsNum[ x ] = low[ x ] = num++;

seen[ x ] = 1;

for( int i = 0 ; i < ady[ x ].size() ; ++i ){

int y = ady[ x ][ i ];

if( !seen[ y ] )

{

printf("%d %d\n" , x , y );

dfs( y , x );

if( dfsNum[ x ] < low[ y ] ){

printf("%d %d\n" , y , x );

}

low[ x ] = min( low[ x ] , low[ y ] );

}

else if( y != parent && dfsNum[ x ] > dfsNum[ y ] ){

printf("%d %d\n" , x , y );

if( low[ x ] > dfsNum[ y ] ) low[ x ] = dfsNum[ y ];

}

}

}

int main(){

int E , u , v , q = 1;

while( scanf("%d %d" , &V , &E ) , V | E ){

while( E-- ){

scanf("%d %d" , &u , &v );

ady[ u ].push\_back( v );

ady[ v ].push\_back( u );

}

memset( seen , 0 , sizeof( seen ) );

num = 1;

printf("%d\n\n", q++ );

dfs( 1 , -1 );

printf("#\n");

for( int i = 1 ; i <= V ; ++i ) ady[ i ].clear();

}

return 0;

}

# Coconuts, Revisited

\*\*\*ID: 616

\*\*\*Tipo: Brute Force

#include <bits/stdc++.h>

using namespace std;

int main(){

int i , n , maxi , aux , cnt;

while( scanf("%d" , &n ) , n > -1 ){

printf("%d coconuts, " , n);

maxi = -1;

for( i = 2 ; i <= sqrt( n - 1 ) + 1 ; ++i ){

aux = n;

cnt = 0;

while( cnt != i ){

aux--;

if( aux % i == 0 ){

aux -= aux/i;

cnt++;

}

else{

cnt = -1;

break;

}

}

if( aux % i == 0 ) maxi = i;

}

if( maxi == -1 ) printf("no solution\n");

else printf("%d people and 1 monkey\n" , maxi );

}

return 0;

}

# Secret Research

\*\*\*ID: 621

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

int main(){

int casos;

string s;

scanf("%d",&casos);

while(casos--){

cin>>s;

if(s=="78" || s=="1" || s=="4")putchar('+');

else if(s.substr(s.length()-2,s.length())=="35")putchar('-');

else if(s[0]=='9' && s[s.length()-1]=='4')putchar('\*');

else if(s.substr(0,3)=="190")putchar('?');

putchar('\n');

}

return 0;

}

# 500!

\*\*\*ID: 623

\*\*\*Tipo: Bignum

#include <bits/stdc++.h>

#include<stdio.h>

using namespace std;

const int MAXD = 2568, DIG = 9, BASE = 1000000000;

const unsigned long long BOUND = numeric\_limits <unsigned long long> :: max () - (unsigned long long) BASE \* BASE;

struct bignum

{

int D, digits [MAXD / DIG + 2];

inline void trim ()

{

while (D > 1 && digits [D - 1] == 0)

D--;

}

inline void init (long long x)

{

memset (digits, 0, sizeof (digits));

D = 0;

do

{

digits [D++] = x % BASE;

x /= BASE;

}

while (x > 0);

}

inline bignum (long long x)

{

init (x);

}

inline bignum (int x = 0)

{

init (x);

}

inline char \*str ()

{

trim ();

char \*buf = new char [DIG \* D + 1];

int pos = 0, d = digits [D - 1];

do

{

buf [pos++] = d % 10 + '0';

d /= 10;

}

while (d > 0);

reverse (buf, buf + pos);

for (int i = D - 2; i >= 0; i--, pos += DIG)

for (int j = DIG - 1, t = digits [i]; j >= 0; j--)

{

buf [pos + j] = t % 10 + '0';

t /= 10;

}

buf [pos] = '\0';

return buf;

}

inline bool operator < (const bignum &o) const

{

if (D != o.D)

return D < o.D;

for (int i = D - 1; i >= 0; i--)

if (digits [i] != o.digits [i])

return digits [i] < o.digits [i];

return false;

}

inline bignum range (int a, int b) const

{

bignum temp = 0;

temp.D = b - a;

for (int i = 0; i < temp.D; i++)

temp.digits [i] = digits [i + a];

return temp;

}

inline bignum operator + (const bignum &o) const

{

bignum sum = o;

int carry = 0;

for (sum.D = 0; sum.D < D || carry > 0; sum.D++)

{

sum.digits [sum.D] += (sum.D < D ? digits [sum.D] : 0) + carry;

if (sum.digits [sum.D] >= BASE)

{

sum.digits [sum.D] -= BASE;

carry = 1;

}

else

carry = 0;

}

sum.D = max (sum.D, o.D);

sum.trim ();

return sum;

}

inline bignum operator - (const bignum &o) const

{

bignum diff = \*this;

for (int i = 0, carry = 0; i < o.D || carry > 0; i++)

{

diff.digits [i] -= (i < o.D ? o.digits [i] : 0) + carry;

if (diff.digits [i] < 0)

{

diff.digits [i] += BASE;

carry = 1;

}

else

carry = 0;

}

diff.trim ();

return diff;

}

inline bignum operator \* (const bignum &o) const

{

bignum prod = 0;

unsigned long long sum = 0, carry = 0;

for (prod.D = 0; prod.D < D + o.D - 1 || carry > 0; prod.D++)

{

sum = carry % BASE;

carry /= BASE;

for (int j = max (prod.D - o.D + 1, 0); j <= min (D - 1, prod.D); j++)

{

sum += (unsigned long long) digits [j] \* o.digits [prod.D - j];

if (sum >= BOUND)

{

carry += sum / BASE;

sum %= BASE;

}

}

carry += sum / BASE;

prod.digits [prod.D] = sum % BASE;

}

prod.trim ();

return prod;

}

};

bignum dpF[1000];

void fact(){

int i;

dpF[0]=1;

for(i=1;i<=1000;i++){

dpF[i]=dpF[i-1]\*i;

}

}

int main(){

fact();

int n;

while(scanf("%d",&n)!=EOF){

printf("%d!\n%s\n",n,dpF[n].str());

}

return 0;

}

# CD

\*\*\*ID: 624

\*\*\*Tipo: Backtracking

#include <bits/stdc++.h>

#define MAX 32

int N , k, maxi;

int seen[ MAX ], resp[ MAX ], a[ MAX ];

void solve( int len , int u ){

if( u > N )return;

if( u > maxi ){

maxi = u;

for( int i = 0 ; i < MAX ; ++i ){

resp[ i ] = seen[ i ];

}

}

for( ; len < k ; ++len ){

seen[ len ] = 1;

solve( len + 1 , u + a[ len ] );

seen[ len ] = 0;

}

}

int main(){

while( scanf("%d" , &N ) == 1 ){

scanf("%d" , &k );

for( int i = 0 ; i < k ; ++i ){

scanf("%d" , &a[ i ] );

}

maxi = 0;

memset( seen , 0 , sizeof( seen ) );

solve( 0 , 0 );

for( int i = 0 ; i < k ; ++i ){

if( resp[ i ] == 1 ){

printf("%d " , a[ i ] );

}

}

printf("sum:%d\n" , maxi );

}

return 0;

}

# Ecosystem

\*\*\*ID: 626

\*\*\*Tipo: Brute Force

#include <bits/stdc++.h>

using namespace std;

#define pb push\_back

#define mp make\_pair

#define x first

#define y second

#define pii pair<int , int >

#define psi pair<string , int>

#define all( x ) x.begin(),x.end()

int MIN( int a , int b ){ return ( a < b ) ? a : b; }

int MAX( int a , int b ){ return ( a < b ) ? b : a; }

string toStr(int n){string s;ostringstream buffer;buffer<<n;s=buffer.str();return s;}

int toInt(string str){int n;istringstream buffer(str);buffer>>n;return n;}

long long pot(long long a,long long b){if(!b)return 1;if(b&1)return a\*pot(a\*a,b>>1); else return pot(a\*a,b>>1);}

int n;

#define ppi pair< int , pair< int , int > >

#define MAX 105

struct Data{

int x , y , z;

Data( int xx , int yy ,int zz ) : x(xx) , y( yy) , z(zz) {}

Data(){}

};

bool cmp( Data d1 , Data d2 ){

return ( d1.x < d2.x || d1.x == d2.x && d1.y < d2.y || d1.x == d2.x && d1.y == d2.y && d1.z < d2.z );

}

int ady[ MAX ][ MAX ];

int main(){

while( scanf("%d" , &n ) != EOF ){

for( int i = 0 ; i < n ; ++i )

for( int j = 0 ; j < n && scanf("%d" , &ady[i][j] ) ; ++j );

vector< Data > v;

for( int i = 0 ; i < n ; ++i ){

for( int j = 0 ; j < n ; ++j ){

if( ady[i][j] && i != j ){

for( int k = 0 ; k < n ; ++k ){

if( ady[ j ][ k ] && i != k ){

if( ady[ k ][ i ] && i != k && j != k && i != j ){

if( (i < j && j < k) || ( i > j && j > k ) ){

v.pb( Data( i + 1 , j + 1 , k + 1 ) );

}

}

}

}

}

}

}

sort( v.begin() , v.end() , cmp );

for( int i = 0 ; i < v.size() ; ++i ){

printf("%d %d %d\n" , v[i].x , v[i].y , v[i].z );

}

printf("total:%d\n\n" , v.size());

}

return 0;

}

# Passwords

\*\*\*ID: 628

\*\*\*Tipo: Recursion

#include <bits/stdc++.h>

using namespace std;

char rule[ 370 ];

int ceros, final;

string toString( int n ){

stringstream ss;

ss<<n;

string s;

ss>>s;

return s;

}

int getFin( ){

if( ceros == 0 )return 0;

if( ceros == 1 )return 9;

if( ceros == 2 )return 99;

if( ceros == 3 )return 999;

if( ceros == 4 )return 9999;

if( ceros == 5 )return 99999;

if( ceros == 6 )return 999999;

if( ceros == 7 )return 9999999;

if( ceros == 8 )return 99999999;

return 999999999;

}

int main(){

int n , m;

char s[ 205 ][ 400 ];

while( scanf( "%d" , &n ) == 1 ){

for( int i = 0 ; i < n ; ++i ){

scanf("%s" , &s[ i ] );

}

scanf("%d" , &m );

printf("--\n");

while( m-- ){

scanf("%s" , &rule );

int len = strlen( rule );

ceros = 0;

for( int j = 0 ; j < len ; ++j ) if( rule[ j ] == '0' )ceros++;

final = getFin( );

for( int q = 0 ; q < n ; ++q ){

string number = "0";

int num = 0;

while( final >= num ){

int len = number.length();

int l = strlen( rule );

string resp = "";

for( int i = 0 , j = ceros ; i < l ; ++i ){

if( rule[ i ] == '#' ){

resp += s[ q ];

}

else if( rule[ i ] == '0'){

if( j <= len ){

resp += number[ len - j ];

}

else resp +="0";

j--;

}

}

printf("%s\n" , resp.c\_str() );

num++;

number = toString( num );

}

}

}

}

return 0;

}

# Self Numbers

\*\*\*ID: 640

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

#define MAX 1000000

bool a[MAX + 1];

int main(){

memset( a, true, sizeof(a) );

for(int i = 1; i <= MAX; i++){

int sum = 0, n = i;

while(n != 0){

sum += n % 10;

n = n / 10;

}

sum += i;

if(sum <= MAX)

a[sum] = false;

}

for(int i = 1; i <= MAX; i++)

if(a[i])

cout << i << endl;

return 0;

}

# Word Amalgamation

\*\*\*ID: 642

\*\*\*Tipo: Strings

#include <bits/stdc++.h>

using namespace std;

#define MAX 105

int len;

bool Valid( const char \*s1 , const char \*s2 ){

int a[ 27 ] , l1 = strlen( s1 ), l2 = strlen( s2 );

memset( a , 0 ,sizeof( a ) );

for( int i = 0 ; i < l1 ; ++i )a[ s1[ i ] - 'a' ]++;

for( int i = 0 ; i < l2 ; ++i )a[ s2[ i ] - 'a' ]--;

for( int i = 0 ; i < 27 ; ++i )if( a[ i ] != 0 )return false;

return true;

}

int main(){

char s[ 7 ];

len = 0;

vector<string> v;

while( scanf("%s" , &s ) && strcmp( s , "XXXXXX" ) != 0 ){

v.push\_back( s );

}

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

len = v.size();

bool b;

while( scanf("%s" , &s ) && strcmp( s , "XXXXXX" ) != 0 ){

b = false;

for( int i = 0 ; i < len ; ++i ){

if( Valid( v[ i ].c\_str() , s ) ){

printf( "%s\n" , v[ i ].c\_str() );

b = true;

}

}

if( !b )printf("NOT A VALID WORD\n");

printf("\*\*\*\*\*\*\n");

}

return 0;

}

# Inmediate Decodability

\*\*\*ID: 644

\*\*\*Tipo: Ad hoc

#include <bits/stdc++.h>

using namespace std;

char code[ 10 ][ 15 ];

int len;

bool isPrefix( char \*s1 , char \*s2 ){

int l1 = strlen( s1 ) , l2 = strlen( s2 ), lon = min( l1 , l2 );

for( int i = 0 ; i < lon ; ++i ){

if( s1[ i ] != s2[ i ] )return false;

}

return true;

}

bool Solve(){

for( int i = 0 ; i < len ; ++i ){

for( int j = 0 ; j < len ; ++j ){

if( i == j )continue;

if( isPrefix( code[ i ] , code[ j ] ) ){

return false;

}

}

}

return true;

}

int main(){

len = 0;

int q = 1;

while( scanf("%s" , code[ len ] ) != EOF ){

while( 1 ){

if( code[ len ][ 0 ] == '9' )break;

scanf("%s" , code[ ++len ] );

}

printf("Set %d " , q++ );

if( Solve() ){

printf("is immediately decodable\n");

}

else printf("is not immediately decodable\n");

len = 0;

}

return 0;

}