CHALLENGE24
INTERNATIONAL 24-HOUR PROGRAMMING CONTEST

Electronic Contest

15th International 24-hour Programming Contest

http://ch24.org

DIAMOND GRADE SPONSORS

fornax

Google

ORGANIZER

mave

Magyar Villamosmérnök- és Informatikus-hallgatók Egyesülete

eestec

LC Budapest

PROFESSIONAL PARTNERS

hte

HÍRKÖZLÉSI ÉS INFORMATIKAI TUDOMÁNYOS EGYESÜLET

NJSZT

Electronic Contest

Welcome to the qualifying round of the 15th International 24-hour Programming Contest!

This document is the problem set for the Electronic Contest to be held on February 28th, 2015.

Rules

The Electronic Contest contains multiple problems. You have all the time in the world to solve them, but we take submissions from 10:00 to 15:00 CET. The inputs of the problems can be found in a zip file that you have probably already downloaded from the website. Most problems will have 10 test cases.

You can use any platform or programming language to solve the problems. We are interested only in the output files, you don't need to upload the source code of the programs that solved them. Once you are done, you can upload your output files via the submission site: http://sub.ch24.org/sub/. Your solutions will be evaluated on-line.

Problems are scored in three major ways:

- **Time** scoring: these problems have exact solutions. When submissions to these are evaluated, a final score is given immediately. From one team, only one correct submission will be accepted for each input (since the input is either solved or not). Given score decreases with time until the end of the contest, so faster solutions get more points.
- **Competitive** scoring: problems that do not have a known "best" solution. Outputs for these problems compete against each other, and scores are scaled according to the best uploaded output. A team may submit multiple correct submissions to one input (only the latest submission will be taken into consideration).
- **Proportional** scoring: solutions to these problems will be compared against a chosen standard. The final score is calculated from the ratio of the evaluated score of the output to a selected constant. A team may submit multiple correct submissions to one input (only the latest submission will be taken into consideration).

Note that points are awarded per output file and not per problem. If your solution only works for some of the input files, you will still be awarded points for the correct output files. A single output file however is either correct or wrong - partially correct output files are not worth any points.

Additional information for time scoring:

Be quick about uploading the output files, because the scores awarded for every output file decrease with time. Uploading it just before the end of the contest is worth **70%** of the maximum points achievable for the test case. During the contest its value decreases linearly with time. However you should also be careful with uploading solutions. Uploading an incorrect solution is worth **-5** points. This penalty is additive, if you upload more incorrect solutions, you will receive it multiple times. For some problems, we distinguish format errors (unparsable outputs) from incorrect outputs, and the former will not be penalised.

Please note that for time-scored problems there is no point in uploading another solution for an already solved testcase because you cannot achieve more points with it. Therefore the system will not register additional uploads for solved testcases for those tasks.

For some time-scored problems, after submitting an incorrect solution, there may be a certain short delay (a couple of minutes) until you can re-submit an updated solution. The delay is applied per team per task per input, and is reported on the submission web interface.

Additional information for competitive scoring:

In this case there will be no score penalty for uploading a solution later, so you are able to achieve the maximum amount of points by submitting in the very last minute - if you beat the other teams' solutions, that is. However, to avoid overloading our server, after submitting a correct solution, you may not re-submit an updated solution for a certain short delay (a couple of minutes). The delay is applied per team per task per input, and is reported on the submission web interface.

Scores for competitively scored problems are recalculated occasionally (every few minutes). Your points may decrease in time (when another team submits a better solution than yours).

Please be aware that only your last submission is considered - not your best one.

Good luck and have fun!

About the Submission site

The location of the submission site is:

http://sub.ch24.org/sub/

You will be able to log in to the submission site with your registered team name and password. After login you can access three main views:

Team Status

You can see your team's status here, with all your submissions and the points received for them.

Submit

This is where you can post your solution files. You can upload multiple output files for multiple problems with a single submit. The naming of the output files must strictly match the following format: X99.out - where X is the problem's character code followed by a number (1 or 2 digits) identifying the test case.

Scores

Here you can see the current standings of the contest. This will not be available in the last hour.

Contact

You can contact us at the email address info@eestec.hu (please include "15ch24" in the subject).

During the contest we will be available on IRC on the `irc.ch24.org` server (using the default port, `6667`), on the following channels:

- `#challenge24` for general discussion about the contest,
- `#info` for a full summary of announcements (read-only),
- dedicated channels `#a`, `#b`, `#c`, `#d`, `#e`, `#f` for problem specific questions.

Note: **all relevant questions/answers will be copied to #info**, which will be also available on the submission site.

Movies!

This year’s tasks all revolve around the glamorous and magical world of cinema.

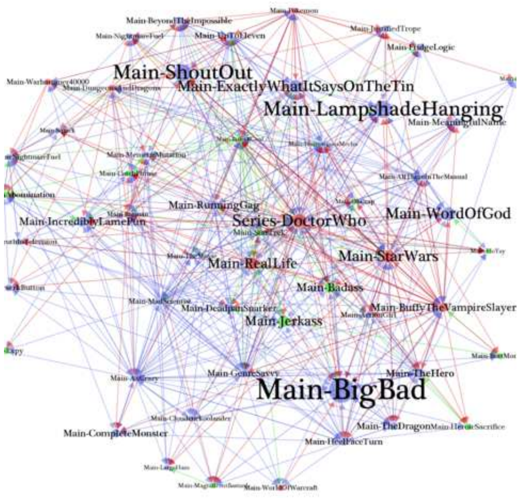
Task Summary

Task	Score	Scoring type	Wrong answer penalty	Delay	Input format
A. Tropes	1000	time	-5 points	0	text
B. MicroPop	1000	time*	-5 points	0	wav
C. Boredom	1000	time	-5 points	0	text
D. Timetable	1000	time	-5 points	0	text
E. Frets	1000	competitive	0 points	30s	png
F. Fast and Furious	1000	time	-5 points	0	text

- * Task B is special: see task description for scoring.
- Wrong answer penalty: Penalty after each wrong output submitted.
- Delay: Time duration until no solution can be submitted for the same input.

For each task there is a dedicated irc channel for questions: `#a`, `#b`, `#c`, `#d`, `#e`, `#f`.

A. Tropes



Every film is built from sequences of "tropes" or "cliches". Often there are usual subsequences, where these tropes follow each other in a particular order (although sometimes unrelated tropes are inserted between members). Given some film scripts (simply given as a series of trope code numbers), determine how long is the longest common subsequence that occurs in every script (allowing for some unrelated tropes in the chain).

Input

First line is the number of test cases in the current file.

Each test case is described thus:

- Number of series in this test case
- For each series, the length
- Then all member numbers of the series

(There is no connection between the test cases.)

Output

Output should contain an integer for each testcase seperated by a linebreak. Each integer should be the exact solution for each case: the length of the longest common subsequence.

Example input

```
1
2
3
10 11 12
2
10 12
```

Example output

```
2
```

The longest common subsequence is "10 12", so the solution is 2.

B. MicroPop

http://hu.wikipedia.org/wiki/Pattogatott_kukorica

Let’s rock? MicroPop popcorn! Do you like it? It’s a final countdown! Some exciting movie for the evening? Sounds good? Some coke and some popcorn are essential!

Count the popped corn!

Input

Audio is given as 16 bit, mono wav files, sampled at 22050 Hz in different kitchen.

Output

Number of popped corn.

Example input

```
Please refer to 0.wav
```

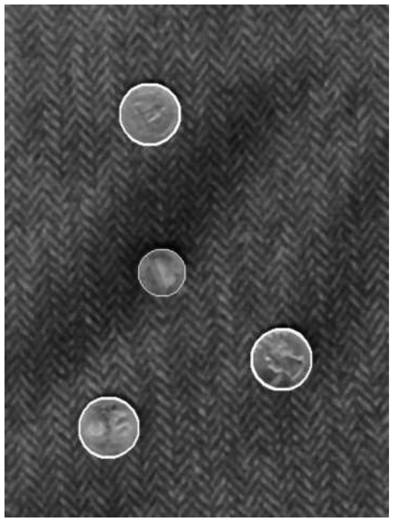
Example output

```
2
```

Scoring

A result within +/- 5% gives you 100 points per input. From +/- 5% to +/- 20% the awarded score drops to zero (over a linear scale) - and you may not submit again for that input! Outside +/- 20% the submission is considered incorrect (resulting in -5 points penalty, and you may submit the input again).

C. Boredom



Some movies are terribly boring. You just have to do something, and you noticed that a 2 Euro coin in your pocket would just fit the circular pattern on the seat of the tall, constantly texting guy sitting in front of you.

Given a set of points, the X coordinates and Y coordinates separated and shuffled (so you don't know which X belongs to which Y), decide whether it's possible to arrange the coordinates together so they fit on a circle (of a finite, non-zero radius).

Input

- Line 1: number of test cases
- Line 2: number of coordinates for test case 1
- Line 3: x coordinates for test case 1
- Line 4: y coordinates for test case 2
- Line 5: number of coordinates for test case 2
- etc.

Output

- Line 1: answer for test case 1, "yes" or "no" (without quotes)
- Line 2: answer for test case 2.
- etc.

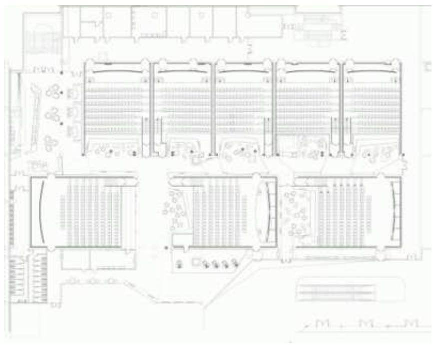
Example input

```
3
3
3.01000000 4.01000000 5.01000000
3.05210000 4.05210000 5.05210000
4
1.21875432 4.00031472 12.00000000 132.12605874
0.00000000 0.00000000 0.00000000 0.00000000
5
1.12345678 1.12345678 1.12345678 1.12345678 1.12345678
13.01010101 13.01010101 13.01010101 13.01010101 13.01010101
```

Example output

```
YES
NO
NO
```

D. Timetable



The cinema displays a timetable, showing which movies show on which screens at what time. Because of administrative reasons, there are some weird constraints on where and when the movies can be shown. The manager of the cinema isn't even sure the timetable for the next day can be organized in a way that satisfies all constraints! He turned to you for help.

The timetable is a table, where each row is a 2 hour block, and each column represents one of the screens. For each time block, there are a number of movies they want to show. One movie can be shown on several screens, but only consecutive screens (that is, screens that are neighboring columns in the timetable). There can be screens that don't show any movies in a time block. The cinema always has enough screens, there is no limit on that. One screen can only show one movie in one time block.

The extra constraints are on which movies can follow each other on the same screen. For each movie *m*, there is a list of companion movies from the previous time block that it can follow on the same screen. For each of the companion movies on this list, there must be at least one screen where *m* follows it. All the screens that show *m* must either show one of the companion movies in the previous time block, or be empty. Each movie must be shown on at least one screen.

Tell them if it's impossible to make the timetable this way, or if it's possible, tell them one way the movies can be distributed in the last time block.

Input

One input file will contain several test cases. The first line contains one number, *T*, the number of test cases. Then *T* test cases follow in the following format.

The fist line of a test case contains one number, *B*, the number of time blocks in that test case. Then, for each of the *B* blocks, the movies are described in the following way.

The first line of a time block contains one number, *M*, the number of movies to be shown in that time block. Each movie is described on one line, first is the number *L*, which is the length of the list of companion movies. The rest of the line is *L* numbers, the indices of the companion movies from the previous time block.

Output

Each test case should correspond to one line of output. If the timetable is impossible to make, write "impossible", without quotes, on the line. Otherwise write one possible ordering of the movies in the last time block. That is, the order in which they would appear in the last row of the timetable, separated by spaces. (Each movie in the last time block will appear exactly once, because they have to be shown only on consecutive screens.)

Example

In the example there are 2 test cases. For the first case, one way to arrange the table is like this:

- In the first time block, movie 0 is on screens 1 and 2, movie 1 is on screens 3 and 4
- In the second time block, movie 0 is on screens 2 and 3, movie 1 is on screen 0 and movie 2 is on screen 4

The second test case is similar, but there is a movie in the third time block. There is no way to show it, because it would need to follow two movies that were shown on the two ends of the cinema, but it can't follow the movie that was shown in the middle.

Example input

```
2
2
2
0
0
3
2 0 1
1 0
1 1
3
2
0
0
3
1 0
2 0 1
1 1
1
2 0 2
```

Example output

```
1 0 2
impossible
```

E. Frets

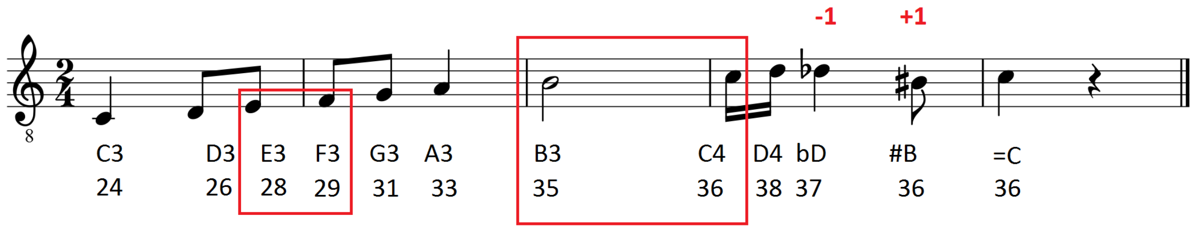
Films have music, and the soundtrack from our favourite film uses only guitars. We'd like to play the sheet music for this film.

Determine the optimal order of hand movements to play this sheet music on a guitar, using the least amount of energy possible.

In the western 12 tone system, the following notes (12 "semitones") exist in a single octave:

```
<<-  C C# D D# E F F# G G# A A# B   ->>
                equivalent to:
<<-  C Db D Eb E F Gb G Ab A Bb B   ->>
```

Meaning that, in the sheet music system, `C#` (C sharp) is the same semitone as `Db` (D flat). The 12 tones then repeat for higher notes (octaves). The first (lowest) set is C1, C#1, D1 etc; then an octave higher, C2, C#2, D2 and so on. We numbered these notes, calling C1 -> 0, C2 -> 12, Db2 == C#2 -> 13, E2 -> 16, etc.



Guitars have six strings. Each string has a base note ("keynote"). It plays this note if you pick it without holding down a fret. Frets are the regularly placed ribs along the guitar's neck. Holding down each fret shifts the played note by a semitone. Frets are numbered from the tuning nut upward (the end of the neck, opposite from the guitar's body), starting from fret 1. "0" means no fret is held down.

You can hold down up to 4 strings simultaneously (using 4 fingers). From the base position of the hand, fingers may be shifted by up to 2 frets (offset: 0..2).

Initially, the hand is positioned over fret 1. It costs 5 units of energy to move the hand by one fret distance.

Additionally, for each played chord, holding down fingers cost energy:

- 1 unit of energy to hold a string to the fret under the hand (offset 0)
- 2 units of energy to hold a string to the fret next to the hand (offset 1)
- 3 units of energy to hold a string to the fret one over (offset 2)
- it doesn't cost energy to just pluck the strings (it's done by a different hand!)

The standard tuning of the guitar is: E2 A2 D3 G3 B3 E4, from the lowest sounding string to the highest. (The strings are tuned a 5 semitones apart, with the exception of the G and B-strings.) You can adjust the tuning of the guitar to achieve optimal play.

Input

Sheet music in PNG files.

Output

On the first line of the output file, the guitar tuning. For example:

```
16 21 26 31 35 40
```

This stands for the default E2 A2 D3 G3 B3 E4.

One line for each chord should follow, with 7 numbers. The first number is the hand position, the following six numbers are for the six strings. For example:

```
1 -1 0 -1 1 -1 -1
```

The hand position is given as a fret index, starting with 1. For each string, the possible values are:

- -1: the string is not plucked
- 0: the string is plucked, but not held to a fret
- 1: the string is plucked, with a finger holding it to the fret under the hand (offset 0)
- 2: the string is plucked, with a finger holding it to the fret next to the hand (offset 1)
- 3: the string is plucked, with a finger holding it to the fret one over (offset 2)

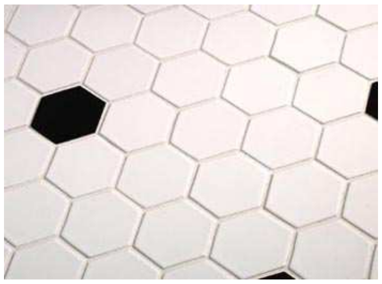
Example input

```
0.png
```

Example output

```
16 21 26 31 35 40
1 -1 3 -1 -1 -1 -1
2 -1 -1 -1 0 3 -1
```

F. Fast and Furious



The movie was very long and we feel like getting home as soon as possible. Therefore we decide to avoid using the brakes unless absolutely necessary.

There is an infinite grid of hexagonal fields. There is a set of grid fields that are designated "walls", and another set that are designated "goals"; the rest are empty. The objective is to drive our vehicle from the start field to any "goal" field, while avoiding the "wall" fields, in the shortest amount of time (least number of steps) possible. Additionally, the vehicle must stop (finish at speed 0) when reaching the goal.

Our vehicle has:

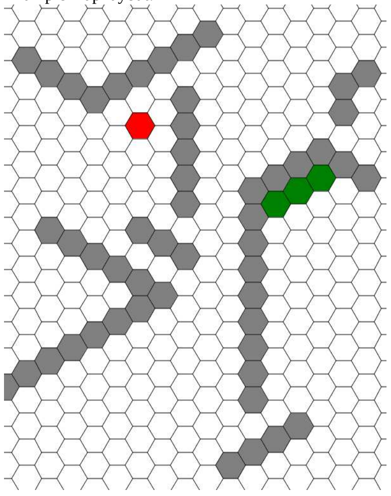
- A current field - initially the field 0, 0
- A current direction - one of `N, NE, SE, S, SW, NW`; initial value chosen in solution
- Speed - integer, >=0, initially 0

In each step, the vehicle executes one of the following commands:

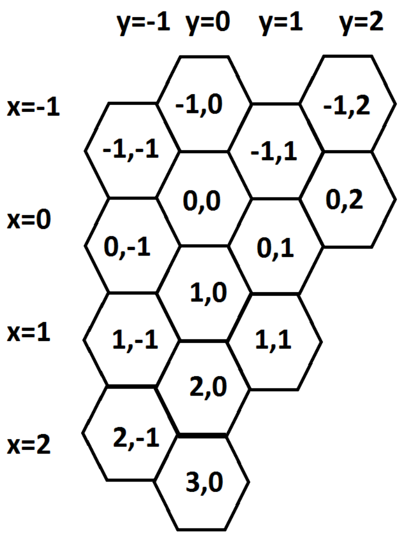
- `F` : go faster; increase speed by 1
- `S` : go slower; decrease speed by 1
- `L` : turn left, not allowed if speed is 0; [N, NE, SE, S, SW, NW] -> [NW, N, NE, SE, S, SW]
- `R` : turn right, not allowed if speed is 0; [N, NE, SE, S, SW, NW] -> [NE, SE, S, SW, NW, N]

After executing the command, the vehicle proceeds in the current direction, going as many fields as the current speed. The vehicle is not allowed to enter a "wall" field at any point while doing this (it's not a jumping car).

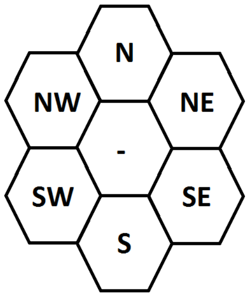
Example map layout:



Coordinate system:



Directions:



Input

- Line 1: number of wall straights
- For each wall straight, a line with four numbers: `x1 y1 x2 y2`. All fields connecting the two specified fields in a straight line (including endpoints) are walls.
- One goal straight, a line with four numbers: `x1 y1 x2 y2`. All fields connecting the two specified fields in a straight line (including endpoints) are goal fields. The vehicle may finish on any of these goal fields.

Output

- Line 1: initial direction
- Line 2: number of time units in solution
- Line 3: the solution; sequence of vehicle commands separated by spaces

Example input

```
3
1 0 2 2
2 2 -1 2
-1 2 -2 0
0 3 -1 5
```

Example output

```
SW
8
F L F L L L S S
```