Preliminary Electronic Contest

14th International 24-hour Programming Contest

http://ch24.org

Preliminary Electronic Contest

Welcome to the testing round of the 14th International 24-hour Programming Contest!

This document is the problem set for the Preliminary Electronic Contest to be held on February 1st, 2014.

The PreEC provides a way for teams to familiarize themselves with our submission system and the general atmosphere of the competition. Whether teams participate in this testing round or not, or whatever results they achieve, will not have any consequences later in the competition.

The three problems we selected for the PreEC may not be the kind of problems you would consider especially challenging, however they provide some clues about the Electronic Contest - the required tools, the usage of our submission system and so on.

Rules

The Preliminary Electronic Contest contains three problems. You have all the time in the world to solve them, but we take submissions from 9:00 to 21:00 CET. The inputs of the problems can be found in a zip file that you have probably already downloaded from the website. Each problem will have exactly 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 should subscribe to the public mailing list at http://lists.ch24.org to receive announcements and to be able to send feedback. The address of the list is `ch24@ch24.org`.

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),
- `#p`, `#q`, `#r` for problem specific questions.

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

Prologue

You have recently discovered the real purpose of life, which is to run a network of cinemas. You immediately gave up your previous plans (of running a telecommunications carrier), and bought a couple of cinemas. The business is booming, and you've already found three areas where you can use your background in programming to build a competitive advantage.

Task Summary

Task	Score	Scoring type	Wrong answer penalty	Delay	Input format
P. Film Logistics	1000	time	-5 points	0	text
Q. Soundtrack vs. popcorn	1000	time	-5 points	0	wav
R. Seating order	1000	competitive	none	120s	png

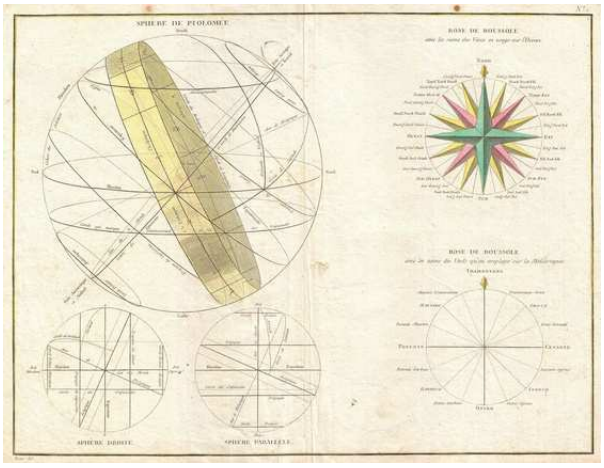
- Wrong answer penalty: Penalty after each wrong output submitted.
- Delay: Time duration until no solution can be submitted for the same input.

For each task there is a dedicated irc channel for questions: `#p`, `#q`, `#r`.

P. Film logistics

You reckon most movies are leaked to torrent sites early on is because someone copies the film when it's being sent to a cinema. To minimize this risk, you buy only one copy for your cinema network, and after playing it in one cinema you send it to the next cinema right away. You always use the shortest path between two cinemas. There's a rotation plan, so every cinema has exactly one movie to play (movies are all the same length anyway). This way at least you know who is to blame when the first torrent appears.

The question is: how much does a movie travel until it gets back to the first cinema? (This is proportional to the risk that it ends up on torrent sites.)



source: http://commons.wikimedia.org/wiki/File:1775_Bonne_Map_or_Chart_of_the_Spheres_and_Compass_Rose_-_Geographicus_-_Spheres-bonne-1775.jpg

Input

A closed path on the surface of the Earth is given with a sequence of latitude and longitude coordinates. The first line is an integer N, the number of cinemas. The following N lines are lat;lon coordinates of each cinema in the order they are visited. (After the last cinema the movie returns to the first one). Earth is a perfect spherical planet with radius 6371 km.

Output

The total travelled distance of a movie in km to at least meter precision.

Example input

```
3
10 40
-55.5 111.11
-18 -70
```

Example output

```
34059.938559
```

Q. Soundtrack vs. popcorn

A well-composed soundtrack of a movie is in a very close relationship with the message of the movie. In fact, they are so close, that careful examination of the soundtrack may reveal important charasteristics about the movie.

source: http://indianapublicmedia.org/arts/bankruptcy-hits-big/

Lately new trends appeared in music used for movies. Followers of the minimalistic genres use clean sine waves instead of inherently complex musical instruments. British researchers also found that using frequencies divisible by 100 between 500 Hz and 10000 Hz yield the best result in popcorn sales. Other researchers confirm that the number of tones in the soundtrack at any given moment is proportional to how fascinating the scene is.

You want to estimate the maximum popcorn flow for a movie - which is (as the above research indicates) proportional to the maximum number of tones audible simultaneously.

Input

The soundtrack is given as an 8 bit, mono wav sampled at 44100 Hz, containing a (rough) mix of different, n*100 Hz sine waves (5 <= n <= 100). Tones are always at least 1.2 seconds long.

Output

Two integers N and T separated by a space; N is the number of tones and T is the time in seconds (from the beginning of the sound track) when the movie is the most fascinating. If there are more, equally fascinating moments and/or these moments are more than 1 second long, any such moment is accepted as valid output.

Example input	Example output
Please refer to 0.wav.	`2 5` (but for example 2 4, 2 6, 2 10, 2 11, etc. are also accepted).

R. Seating order

Despite of all the recent efforts for standardization, people are still being born with different genomes, resulting in different sizes. The relevance to the world of movies is that tall people should not sit before short people, because they may block the view.

Because the necessary technology is still rather expensive, it's not feasible yet to do full genetic screening of every guest when they purchase a ticket. Therefore the seating order must be resolved right before the movie starts.

There is a common "best practice" method that provides good results using simple heuristics, while avoiding chaos. People first sit where their ticket says so; then if someone doesn't see the screen, they ask the person sitting in front of them to swap seats. Because most people are polite and will not climb over a row of seats, they usually stand up and move along the rows and aisles. Customers sitting in the same row on adjacent seats may also swap seats, if they think this can help the process along. To avoid confusion and mass panic, every swap is fully finished before a new one can be started.

source: http://thegraphicsfairy.com/funny-old-photo-tall-man-with-short-man/

This moving around of viewers is interesting to you since it takes a lot of time (longer than the actual movie), and you can show commercials during the process. It is essential to be able to estimate the amount of time you have to fill with commercials - which will be proportional to the number of swaps. To get a clearer picture on how the process works, you sample 10 audiences, and attempt to determine how the shuffling will happen.

Data is stored in greyscale PNG files; the darker a pixel is, the shorter is the customer sitting there. You need to find a reasonably short sequence of swaps that would result in clear view for everyone. Your solution will most probably be more efficient (smaller number of swaps) than what the guests would do in reality, so it will provide a good minimum estimation on how much time you can contract for commercials.

Scoring

For each given input, submissions from all teams are evaluated; The score is based on the number of swaps compared to the best submission (less is better, too many swaps means 0 score):

```
SCORE = 100*(1 - sqrt(1 - BEST/SWAPS))
```

Input

Smallish PNG files containing only grey, black and white pixels. Color space of the input may be greyscale or 1 bit black & white or indexed, but in any case actual pixels will be at most "256 shades of grey" between 0 (black) and 255 (white).

Output

First line is an integer that describes the number of swap sequences. Each subsequent line is a swap sequence, a list of integers separated by spaces, starting with three numbers L, X and Y. L is the number of swaps, X and Y are the reference coordinates assuming the top-left pixel is at x=0;y=0 and X is the horizontal axis. The rest of the line contains swaps. Each swap is described by two numbers: a step direction, which may modify the reference coordinate by one and a neighbor direction that addresses the other pixel the current one is swapped with. Stepping (base coordinate modification) happens before swapping. Swap sequences and swaps within a sequence are executed in order of appearance. Directions are:

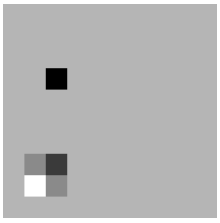
value	description	relative address
0	stay in place	x:0 y:0
1	north	x:0 y:-1
2	east	x:+1 y:0
3	south	x:0 y:+1
4	west	x:-1 y:0

For example a swap line "2 6 8 0 3 1 2" would first swap pixel x=6;y=8 with pixel x=6;y=9 then pixel x=7;y=8 with pixel x=8;y=8, leaving the reference coordinate x=7;y=8 at the end of the sequence.

The screen is on the left; a valid output transforms the input image so that pixel values are in ascending order from left to right in each row. There is no requirement for any sort of vertical arrangement.

Addressing out of the image dimensions for either end of a swap is a fatal error (submission not accepted); swapping a pixel with itself is legal and will not change the image. Integers in any field are between 0 and 2^{30}. The usual *maximum size of submission* may limit the number of swaps a valid output file can contain. Number of swaps (L) is greater than zero.

Example input



Example output

```
5
2 2 3 0 4 4 4
2 2 7 0 4 4 4
1 1 7 0 2
2 2 8 0 4 4 4
7 2 8 0 2 2 2 2 2 2 2 2 2 2 2 2
```