Electronic Contest

12th BME International 24-hour Programming Contest

http://ch24.org

MAIN SPONSOR

SAP

DIAMOND GRADE SPONSORS

BASE 4 FORNAX Google !iFlow

SILVER GRADE SPONSORS

T LogMeIn

INNOVATIVE SPONSOR

iGO My way.

TALENT SPONSOR

Prezi

3D MODELING SPONSOR

GRAPHISOFT

PROFESSIONAL PARTNERS

hte HÍRKÖZLÉSI ÉS INFORMATIKAI TUDOMÁNYOS EGYESÜLET

NJSZT

MEDIA PARTNER

Impulzus

ORGANIZER

mave Magyar Villamosmérnök- és Informatikus-hallgatók Egyesülete www.eestec.hu

SPECIAL THANKS TO

MŰEGYETEM 1782

Budapest University of Technology and Economics
Department of Automation and Applied Informatics

Electronic Contest

Welcome to the qualifying round of the 12th BME International 24-hour Programming Contest!

This document is the problem set for the Electronic Contest to be held on February 25th, 2012.

Rules

The Electronic Contest contains six 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. 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://ch24.org/sub/. Your solutions will be evaluated on-line.

There are two major problem types:

- Non-scaled problems: problems that have an exact solution. 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). In the EC, A, B, C, E, F are non-scaled problems.
- Scaled problems: 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). In the EC, only D is a scaled problem.

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.

To avoid overloading our server, before submitting an updated solution for an input we will apply a delay. The delay is traced per team per task per input.

Additional information for non-scaled problems:

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 the non-scaled 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.

Additional information for scaled problems:

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.

Scores to scaled 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://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.

Task Summary

Task	Scaling	Wrong answer penalty	Delay	Input format
A	No	-5 points	60s	png
B	No	-5 points	60s	text
C	No	-5 points	60s	text
D	Yes	-5 points	60s	text
E	No	-5 points	60s	wav
F	No	-5 points	60s	text

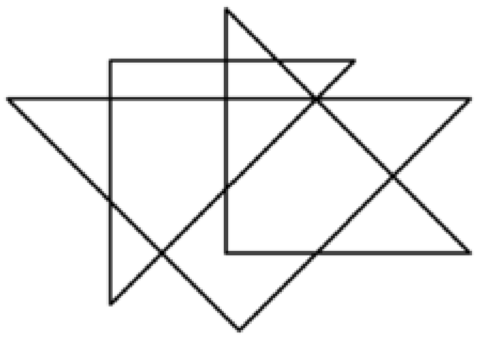
- Scaling: The score for this problem may change over time depending on submissions by other teams. (Note that your last submission is considered and not your best one.)
- Wrong answer penalty: Penalty after each wrong output submitted.
- Delay: Time duration until no solution can be submitted for the same input.

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`). For general discussion about the contest use the `#challenge24` channel, for problem specific discussion use `#a`, `#b`, `#c`, `#d`, `#e`, `#f` channels.

A. Triangles



Your task, assigned by the International Office of Paper Pushing (IOPP), is to count triangles on monochrome images. According to IOPP policy:

- a line is an uninterrupted series of black pixels
- furthermore a line is always horizontal, vertical or at 45 degrees
- a triangle is made up of 3 lines connected in 3 different corner pixels

Input

PNG files.

Output

Exact number of black triangles in the PNG.

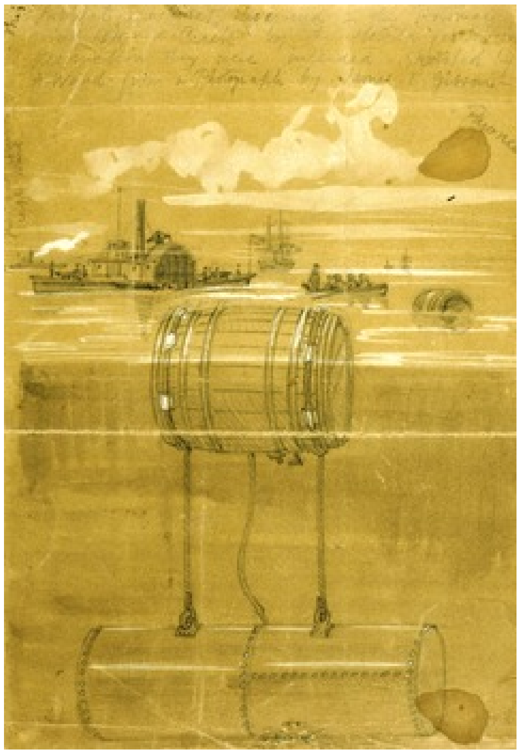
B. Mines

As a member of the Special Operations Forces of a small country, your job is to clear a minefield. Unfortunately, military top brass is rather more concerned with politics then the general welfare of their officers. Being in a sad state of ignorance of how mines (and the bodies of soldiers) actually work, they demand you clear the minefield by putting timed ignition charges on each mine.

This does require you to personally fiddle with armed mines of various sizes but invariable deadliness.

Thankfully, you realize in time that when the primary charge of these mines blows up, the shockwave is usually strong enough to blow up other mines in the vicinity - and the other mines blowing up can blow up mines yet farther away. If you could determine which mines to set off to maximize the cascade effect, you could minimize the number of mines you have to manipulate with your own quite defenseless hands.

In short: given a precise map of the minefield, your task is to eliminate all mines by directly setting off as few mines as possible.


source:
http://en.wikipedia.org/wiki/File:Waud_-_infernal_machines.jpg

Input

```
N
X[0] Y[0] R[0]
...
X[N-1] Y[N-1] R[N-1]
```

`N` is the number of mines, `X[i] Y[i]` is the coordinates of the ith mine and `R[i]` is its detonation radius.

Output

```
M[1] M[2] ... M[K]
```

K is the minimum number of mine detonations, M[i] is the index of the ith detonated mine.

Example input

```
5
-15 4 10
12 4 8
-10 -4 8
2 10 16
18 10 8
```

Example output

```
0 3
```

C. Lumberjack

A lumberjack has to cut down N trees which were planted along a straight line with fixed distance between them.

If the lumberjack cuts down a tree he can make it fall either left or right along the line. A tree can topple another tree if it falls on it and may cause the fall of many other trees (by the domino effect).

Which trees should the lumberjack cut down so that all trees fall with the minimum number of cuts?

source:
http://pixabay.com/en/saw-chainring-chainsaw-chain-wood-5901/

Constraints of the domino effect:

- The height of the ith tree is H[i], 1 <= i <= N
- If tree i falls left then all standing trees j fall left as well for i-H[i] < j < i
- If tree i falls right then all standing trees j fall right as well for i < j < i+H[i]
- A tree can only fall once

Input

```
N
H[1] H[2] ... H[N]
```

Output

```
K
T[1] T[2] ... T[K]
```

K is the optimal number of tree cuts. The absolute value of T[i] is the index of the ith tree that is cut down. (Note that the order in which trees are cut down may matter.) T[i] is negative if tree T[i] falls left.

Example

Input	Output

`5`
`1 2 3 1 1`
	Two of the possible outputs are
	
	`2`
	`3 -2`
	
	or
	
	`2`
	`1 2`

D. If

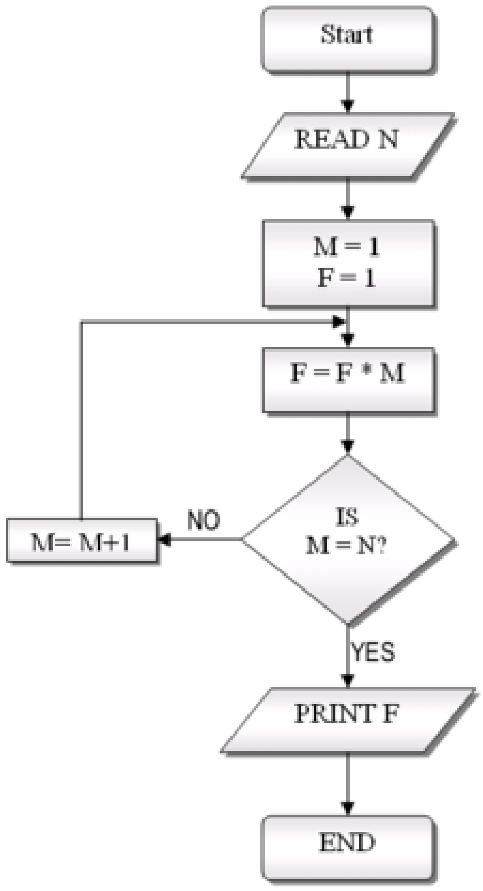
Your task is to optimize the input source code for size. The source code will run on an 8-bit architecture.

There are 256 variables (numbered from 0 to 255), at most 2 inputs (numbered 0 and 1) and 8 outputs (numbered from 0 to 7). All values are 8-bit unsigned integers: values stored in variables, read from input, written to output and existing as constants in the source code. All arithmetics are evaluated modulo 256.

Inputs don't change until the program executes the END instruction. Before running a program, variables and outputs are set to 0.

Input of this task is a source file; output is an optimized version of the source. A submission is not accepted, if:

- it does not give the same output as the original program would, for arbitrary input; or
- it executes more than 10^7 instructions for any input; or
- it references an address out of boundary.



source:
http://en.wikipedia.org/wiki/File:FlowchartExample.png)

Language

R, *A* and *B* are any of the variables, *const* is a constant integer; *input(N)* fetches the *Nth* input; *output(N)* is the *Nth* output; *label* is a jump destination point in the source code marked by an unique integer.

description	**syntax in input/output files**
expressions	
R = const	n,const,R
R = A + B	+,A,B,R
R = A - B	-,A,B,R
R = A > B	>,A,B,R
R = A < B	<,A,B,R
R = A == B	=,A,B,R
R = A != B	!,A,B,R
R = input(A)	i,A,R
conditional jump: IF *R* != 0 GOTO *label*	c,R,label
unconditional jump: GOTO *label*	g,label
generating output: *output(A) = R*	o,A,R
jump *label* (destination of jumps)	l,label
end of execution	e
comment (whole line ignored):	#,text

Scoring

Score is calculated based on the number of instructions of the submission, compared to other teams' submissions - shorter outputs get higher scores.

Example

Original	Optimized
l,1 n,0,2 i,2,4 n,1,1 i,1,3 n,3,2 +,4,2,4 n,4,1 +,3,1,3 n,2,2 +,3,2,6 n,0,1 o,1,6 n,6,2 +,4,2,5 n,1,1 o,1,5 e	n,0,2 n,1,4 n,6,5 n,9,6 i,2,1 i,4,3 +,3,5,0 o,2,0 +,1,6,0 o,4,0 e

E. Beacons

Our researchers have developed an advanced acoustic positioning system. We now have an installed prototype, and the field test is about to begin.

Photo by NASA

The positioning system consists of several rotating towers, deployed at known locations. Each tower has a loudspeaker attached to it, emitting a sinewave sound at a constant frequency. The towers rotate around their vertical axis counterclockwise once a second, while keeping their heading synchronized at all times (for example, they always face north at the same time).

The sound attenuates with distance (because the power of the signal spreads out on a large surface): the measured power is proportional to `1/R^2` and the measured amplitude to `1/R` where `R` is the distance from the tower.

The towers and the rotating mechanisms were very expensive, so we had to choose cheap loudspeakers. They cannot be exactly calibrated to the same loudness. Even so, the generated amplitudes shouldn't differ more than 10% between towers (measured at the same distance and direction).

The speakers are installed in a way that the emitted signal is strongest in the direction of the speaker, with a +/- 45 degree spread.

The prototype installation is situated in a friendly, sheltered valley, with no wind or any source of external noise.

The speed of sound in the local environment was measured to be a stable 340.29 m/s.

Our coordinate system uses the meter for a unit. X and Y are euclidean coordinates. The earth can be considered completely flat in the area of the prototype. The value of X grows from west to east, that of Y grows from south to north.

The towers are described in the `towers` file. Each line in the file gives the position and the emitted frequency of a tower, like so:

```
X Y FREQ
```

Your task is to take a number of sound recordings we made in the field, and in each case determine the position of the recording apparatus. The apparatus was not moved during recording.

Input

The inputs are WAV files - sound recordings made at various positions.

Output

The output should be the position of the microphone:

```
X Y
```

For the field test to be considered successful, the calculated and real position must not differ by more than 2 meters.

Example input

You may use `example.wav` input for calibration.

Example output

Solution for `example.wav`:

```
-179.4 -95.3
```

Currency hack

You end up having a small fortune in your national currency.

You have found a fast way of travelling around in your city, and discovered that there is an uncommonly large amount of independent money exchange offices (most of them opened just in the previous week).

The exchange offices are privately owned, and offer various different exchange rates. You decide to use your small fortune to exploit the exchange rate differences by taking your money around and converting it back and forth between all the different currencies.

Your goal is to finish with the most money possible, in your own currency, by the end of the day.

All money in the city is managed electronically, so there are no rounding rules; you can have and change fractional amounts of money.

Input

First line: *N K S*, where *N* is the number of currencies (currency 0 is your own national currency); *K* is number of exchange offers and *S* is the amount of money you start with.

Next *K* lines are offers in the following format: *A_i B_i R_i M_i*, which means converting from *A_i* to *B_i* at rate *R_i*, at most *M_i* amount of money.

Output

First and only line is the maximum profit that can be realized (0.1% relative error is accepted).

Example

Input	Output

`2 3 1000`
`0 1 0.8 500`
`0 1 0.9 400`
`1 0 1.2 10000`
	`32`

In this example, the first offer means that you can give them money in currency 0, and they give you 0.8 times that much in currency 1. During the whole day, you can give them at most 500.