CHALLENGE24

INTERNATIONAL 24-HOUR PROGRAMMING CONTEST

Electronic Contest

13th International 24-hour Programming Contest

http://ch24.org

MAIN SPONSOR

SAP

DIAMOND GRADE SPONSORS

FORNAX

Google™

skype™

YAHOO!

SILVER GRADE SPONSOR

iGO™ My way.

PARTNERS

Continental

LogMeIn®

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 13th BME International 24-hour Programming Contest!

This document is the problem set for the Electronic Contest to be held on February 23rd, 2013.

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. 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.

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, F, G 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 normal scaled problem.
- In this EC, E is a special problem that's scaled against fixed constants instead of the submissions of other teams. (see the task description for details).

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. The maximum achievable score for each test case is 100 points.

Additional information for non-scaled problems:

Be quick about uploading the output files, because the scores awarded for output files decrease with time. Uploading one 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.

For some non-scaled 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 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. 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 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://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),
- dedicated channels `#a`, `#b`, `#c`, `#d`, `#e`, `#f`, `#g` for problem specific questions.

Note: **all relevant questions/answers will be copied to #info**, which is also available on the web (http://igor2.repo.hu/ch24/info.txt).

Prologue: Recovering From Recession

International Geek Gadgets, Ltd (IGG) is heavily affected by the current economic recession. The CEO, your old friend, is struggling to find a way out. She already identified the seven major areas in the company where optimizing current processes could result in a noticeable increase of efficiency. She has also realized such optimization would require the help of computer programs.

Instead of hiring an expensive analyst, she thought you'd help her on the basis of friendship - and she offered you the 35 years old PDP-11 gathering dust in her basement to sweeten the deal, if everything goes well.

Beside optimizing business processes, she also opted for advanced quality assurance qualification - auditors are scheduled to arrive in five hours and will have unlimited power to screen any activity in the company. After the auditors put their stamp on the papers, it will be impossible to change any process further.

You enlisted the help of two of your even geekier friends to have a chance at meeting the deadline. You have only five hours to solve as many of the seven deficiencies as possible.

Task Summary

Task	Scaling	Wrong answer penalty	Delay	Input format
A. Cutting back middle management	No	-5 points	60s	text
B. Requirements	No	-5 points	60s	text
C. Road roller	Yes	-5 points	60s	text
D. Octal CNC	No	-5 points	60s	png
E. Stack compressor	-	-5 points	60s	text
F. Backup communication	No	-5 points	60s	text
G. Trains	No	-5 points	60s	wav

- 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.
- E is a special problem that's scaled against fixed constants (see the task description for details).

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

A. Cutting back middle management

This is not the first crisis in the history of IGG. Whenever cost reduction was needed in the past, firing employees worked to some degree. This affected workers at the bottom of the food chain more often; managers could generally evade the threat. Over the decades this caused an unusually large amount of middle managers to accumulate. The CEO finally realized there's no other way to balance the structure of the company but to make their positions redundant.

A middle manager is somebody who has subordinates (who might also be managers), but the CEO is not a middle manager. Everyone can have at most M subordinates. If someone is fired, their subordinates are reassigned to their boss. Determine at most how many middle managers can be fired, such that the "at most M subordinates" rule still holds?

http://www.wpclipart.com/clothes/suit/suit_lapel.png.html

Input

First line contains N and M, where N is the number of managers. In the following N lines, the *I*th line contains:

- k_1, the number of non-manager subordinates of the *I*th manager
- k_2, the number of manager subordinates of the *I*th manager
- k_2 numbers, that are the numbers of the manager subordinates of the *I*th manager

It is guaranteed that the 0th manager is the CEO.

Output

An integer: the amount of middle managers that can be fired.

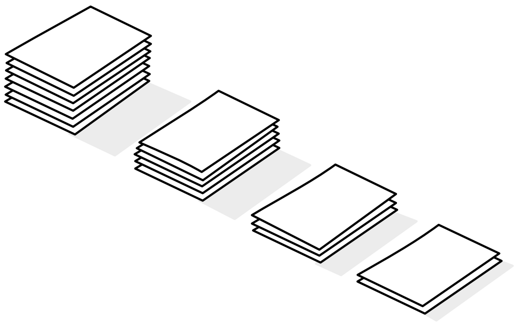
Example input

```
10 4
0 3 6 2 3
2 1 9
2 1 8
2 1 1
4 0
1 0
0 2 7 4
4 0
1 1 5
1 0
```

Example output

```
4
```

B. Requirements


http://www.clker.com/clipart-piles-of-paper.html

Auditors are few, but papers to push around are many. For low level bulk processing, auditors employ secretaries. These secretaries are not very clever (this is why they're not auditors yet), but if a simple task is explained well, they can do it precisely and without asking needless questions (the famous motto for auditor's secretaries is "understanding is not required - only obedience").

One of the important tasks in such audits is to set up procedure *manuals*. Such a *manual* contains (among other things) a list of requirements. Once a process doesn't meet a requirement, the difference must be documented clearly.

To make sure there are enough *manuals* available for describing all processes of a company, the custom is that auditors dictate all *manuals* to secretaries after the audit ends. The secretaries then merge all *manuals* with all other *manuals*. All possible combinations are kept, to be on the safe side; but since this merge produces new *manuals* in the system, they have to restart merging again, producing even more combinations. This goes on and on until the end of the working day. (Precisely until the end: since the labor union of secretaries is strong, they don't have to do any overtime.)

A merge of two *manuals A* and *B* will result in a new *manual C*, and is produced in three steps:

- take an empty sheet and label it as *manual C*
- copy all requirements of *A* to *C*
- append all requirements that are present in *B* but are not present in *A* to the end of *C*.

Secretaries are not just hard-working but are also very pedantic. For the two *manuals* mentioned above, they not only merge *A* and *B* into *C*, but also *B* and *A* into *D*.

Looking at all this, you realize what they did not: each requirement is just a 1-bit property of a *manual* (whether the manual has it or not), since the order of requirements does not really matter. Sooner or later, there are going to be *ultimate manuals* that will contain all requirements ever invented by the auditors. Since you know you will need to prepare for the *ultimate manuals*, the only question remains: how many of those will exist? If only a few, you may sneak in and shred them before anyone notices.

Input

After the end of the audit, there are *N* different requirements the auditors came up with, that occur in the first batch of manuals. The secretaries do the full merge on this batch (merging each with each other, collecting the new manuals in a separate bin not to be mixed into the current merge efforts). Having finished the merge, they take all the old and new manuals together, and start over again, merging everything with everything else. The secretaries have time for doing *T* full iterations until the end of the day (they don't do partial iterations, to avoid confusion).

After *N* and *T*, the input file contains 2^N integers. Integer no. *i* (numbering starts from 0) is the count of how many copies of manual *i* exist (copies are handled like separate manuals when merging, but have the same set of requirements in them). Manual no. *i* lists only those requirements where the binary digit of *i* is 1 (i.e. manual 0 contains no requirements, manual 1 contains only requirement #1, manual 2 contains only requirement #2 while manual 3 contains both requirements #1 and #2).

Output

A single integer, the number of *ultimate manuals* that will exist by the end of the day modulo 1000000007.

Example input

```
2 123
87 65 43 21
```

Example output

```
991654691
```

C. Road roller

One of the factors slowing down operations at the IGG steel mill is the uneven concrete surface of the factory yard where incoming raw material is received and sorted. Fortunately there's still time to use one of the brand new products of the company, the *EZRoller* to get a flat surface. *EZRoller* looks like a traditional road roller but has a number of advanced features:

- no need to roll forth and back over the same spot: a single roll results in a completely flat surface, at the level specified by the user (*EZRoller* is heavy)
- cutting edge 3000 BHP power plant (*EZRoller* is fast)
- there is a button to start, stop, and turn (*EZRoller* is very user friendly)

http://searcharchives.vancouver.ca/
steam-roller-and-crew-at-corner-of-fleming-and-gibson-roads

Unfortunately these features impose a limitation on the path *EZRoller* can follow: it is not possible to turn without stopping, and the turn button allows turning only in 45 degree steps. Due to inertia, stopping and starting are very time consuming operations, so the operator of *EZRoller* is asked to avoid these whenever possible (it's much easier to rely on *EZRoller's* extreme strength and sturdiness, and "think outside the box", so to speak).

You need to find a short path for *EZRoller* for leveling all the points specified on the surface of the factory yard.

Input

Lines of X Y pairs for each point that needs to be levelled (X and Y are integers).

Output

The task is to define a line of connected segments by their endpoints such that it covers all the points given in the input and the number of segments is minimal.

The segments can be vertical, horizontal or 45 degree diagonals. The endpoints of the segments should be given as X Y pairs in order, one per line (X and Y must be integers).

Score

This is a scaled problem and the real score is

```
SCORE := round(100 * (1 - sqrt(1 - Kmin/K)))
```

where K is the number of segments in the output and Kmin is the best submission so far.

Example input

```
1 5
2 2
3 4
4 1
4 4
5 1
5 3
5 5
```

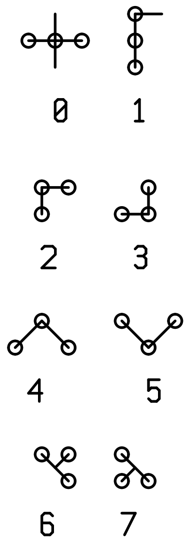
Example output

```
4 1
3 1
3 5
1 5
1 1
5 5
5 1
```

So (4, 1) is connected to (3, 1), (3, 1) is connected to (3, 5) and so on. This gives a path of 6 segments.

D. Octal CNC

IGG uses CNC milling to manufacture some of the parts it makes. The CNC machine was bought as a bare bone construction, which means it came without an external communication port and can be programmed only via a small built-in touch screen. To cut the costs even more, IGG bought the cheapest firmware - the one that doesn't include an advanced user interface. Engineers program the CNC using octal digits drawn as strange glyphs on the display.



canonical form of the digits

The engineers often write down their programs on paper, as that's the only way backing up their data. A lot of valuable CNC programs are stored this way. This is clearly not compliant with the policy the quality assurance auditors will mandate. The only solution is to finally convert the programs into conventional numbers, stored in text files. Unfortunately, due to the amount of data and the short deadline, this is not feasible to do by hand.

The engineers used their off-work hours to scan the papers - everything is now available in high resolution gray scale bitmaps. The only missing step is to process the images and convert the symbols into numbers.

There are 8 digits, from 0 to 7. The canonical printed forms are as given above - hand written symbols may be less perfectly shaped. Each digit consists of 3 circles and 2 lines.

Input

A grayscale png with multiple "handwritten" lines of octal digits (black ink on white paper that may have texture) with the following properties:

- maximum size of a page is 16384*16384 pixels;
- digits may be scaled, aspect may change;
- digits may be rotated randomly, within +-15 degrees;
- spacing between characters are not even, but lines or cirles of different characters never cross or touch;
- lines of text are not always straight;
- each line ends with a checksum digit.

The checksum digit is the sum of digits in the line (without the checksum digit) modulo 8.

Output

Outpout is a stream of digits from '0' to '7' in ASCII text, checksum included. The output shall cosist exactly as many digits as many gylphs are on the input png. The evaluator will ignore empty lines and white space.

Example input

A small "hello world" example is provided as `example.png` among with the input files. File `reference_digits.png` contains a list of digits from 0 to 7, without checksum, written in a relatively clean form (actual input images may have more distortion).

Example output

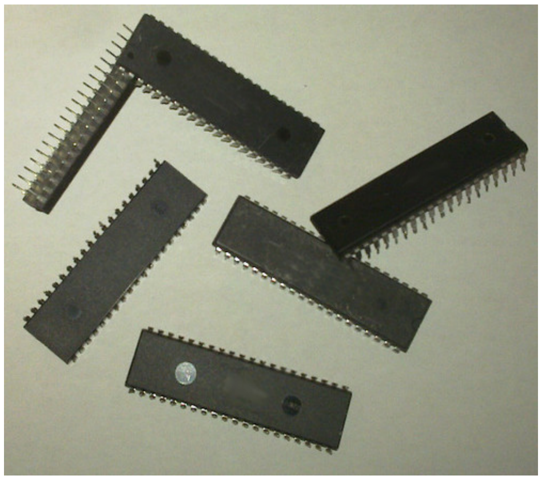
Output of the "hello world" example:

```
0421501451541541570425
0421671571621541440423
```

E. Stack compressor

In its wide range of different products, a flagship model carefully designed and manufactured by IGG is a clever processor. The unit implements a huge flash-based program memory, a large SRAM-based stack, a great 32-bit ALU and rich I/O facilities, all integrated on the same chip. Even though the product itself is one of the best on the market, due to a somewhat unfortunate and ill-planned marketing campaign, sales have lagged. There is still a large quantity left from the first production batch.

The sales department now suggests that demand could be raised by demonstrating the extraordinary capabilities of the device in specific application domains, such as text compression.


The target platform (**illustration** - the actual product may differ in look while retaining size and pin compatibility)

Your task is to show how well the architecture is suited for text compression by providing example programs that output a much larger amount of meaningful data than the original size of the program.

Write program that outputs exactly the input file of the task - the smaller the code is, the better. All instructions increases program size by 1 except that PUSH counts as 2.

Numbers are 32-bit signed numbers ($-2^{31} \le n < 2^{31}$). Overflows cause error. It is an error if there are not enough elements on the stack for an instruction. The stack is empty at startup. Limits:

- max stack size: 20 million elements
- max step count: 300 million instructions executed

Instruction set

Stack description notation:

- "..." means original content of the stack
- "a|b" means *b* on top of *a* (the last is the topmost element on the stack)
- "s[N]" the Nth element of the stack

instruction	**description**	**stack before**	**stack after**
PUSH const	Push a constant to the top of the stack.	...	...|const
OUT	Pop the top element of the stack and write it to the output as an ASCII character. It is an error if *a* > 127 or *a* < 0.	...|a	...
READ	Pop the top element of the stack, *a*, and push a copy of the *a*-th element of the stack on top. For *a* ≥ 0, the stack is indexed from the bottom. For *a* < 0, the stack is indexed from the top. (for example the element below *a* can be indexed with -1).	...|a	...|s[a]
JGZ	Jump if Greater than Zero: Pop the top two elements of the stack, *a* and *b*, and jump *a* instructions if *b* > 0. Jump forward if *a* > 0, jump backward if *a* < 0 (*a* = -1 jumps back to the current JGZ instruction). Execution stops if JGZ jumps out of the code in either direction.	...|b|a	...
ADD	Pop the top two elements of the stack, and push their sum.	...|b|a	...|a+b
MUL	Pop the top two elements of the stack, and push their product.	...|b|a	...|a*b
DIV	Pop the top two elements of the stack, and push their quotient and remainder (*q*=*b*/*a*, *r*=*b*%*a* like in C99).	...|b|a	...|q|r

Input

Plain text 7-bit ASCII, written in English. Expect some control characters (e.g. form feed).

Output

The program in plain text: a list of instructions, each in a separate line, case-sensitive.

Scoring

Scores for this problem are scaled against fixed constants (not the solutions of other teams), otherwise it works like normal scaled problems: multiple solutions can be submitted but only the last one counts.

The real score of a size S program is

```
SCORE = 20 + 80*(A - S)/(A - B)
```

where A, B parameters are given in the table below. SCORE is rounded to the nearest integer and then truncated to [0,100].

input	1.in	2.in	3.in	4.in	5.in	6.in	7.in	8.in	9.in	10.in
A	5500	5280	9080	17900	20000	25500	40000	41000	58000	65000
B	3500	3600	6000	10000	10000	15000	25000	27000	34000	32000

Example input

```
Hello World!
```

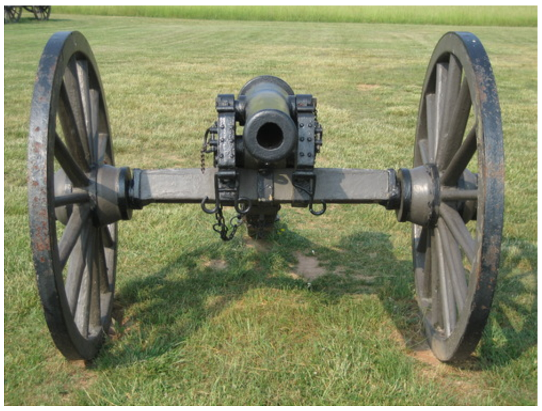
Example output

The size of this program is 50 and it prints the example input including the terminating newline.

```
PUSH 72
OUT
PUSH 101
OUT
PUSH 108
PUSH -2
PUSH -1
MUL
PUSH -2
READ
OUT
PUSH -1
ADD
PUSH -1
READ
PUSH -9
JGZ
PUSH 6700
PUSH 21714
PUSH 22287
PUSH 6511
PUSH 200
DIV
OUT
PUSH -1
READ
OUT
PUSH -8
JGZ
PUSH 15
DIV
ADD
OUT
```

F. Backup communication

At each site, IGG has many buildings, with modern means of communication connecting them. Unfortunately these communication lines all depend on electricity. In case of a blackout, transmitting management orders becomes impossible using those channels. Off the shelf backup communication systems are extremely expensive, and the CEO has realized IGG is already manufacturing all the important components of an alternative solution: cannons, gunpowder, titanium capsules, paper and pen. The idea is to place a single cannon near the HQ and fire orders in the direction of each building. In case of emergency, the typical usage is to send the same message to all buildings with the smallest amount of effort.


http://commons.wikimedia.org/wiki/File:Parrottgun.jpg

Your task is to determine the optimal placement of the cannon (on each of the 10 sites of the company) to minimize the energy required to broadcast orders.

The energy required for a single shot is quadratic in the launch velocity *v* of the capsule, ie.

```
E = const * v * v
```

The elevation and direction of the cannon barrel can be easily set as well as *v*. The broadcast cost is the sum of the energies of all shots with optimal cannon settings.

Careful measurements show that the aerodynamic drag can be discarded and the maximum range of the cannon is larger than the distance between any two buildings.

Input

First line contains N the number of targets, the next N lines contain X,Y coordinates of the target buildings in km.

Output

The X,Y coordinates of the cannon, it must be within 0.1 m of the optimal placement.

Example input

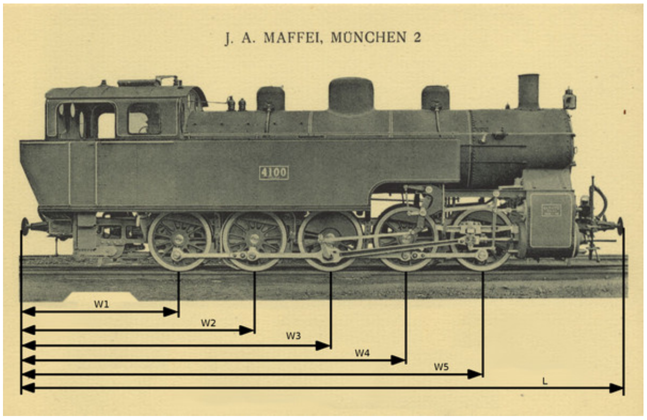
```
5
-3.7 0.5
1.2 7.7
4.8 -7.7
-2.3 -0.9
5.2 3.4
```

Example output

```
-0.912975 0.321035
```

G. Trains

IGG ships its products mainly using cargo trains. At most sites there is a dedicated yard for sorting wagons and loading trains. IGG owns a large fleet of different locomotives and wagons, but their allocation is mostly ad-hoc as no one tries to keep track of individual vehicles entering or leaving a given site. This has to change for the audit.



http://commons.wikimedia.org/wiki/File:Joseph_Anton_von_Maffei_06.jpg

The proper solution would be to install a unique ID tag on each locomotive and car (an RFID tag, a bar code, etc.), however this would take too much time. A much cheaper alternative is to install a microphone at the entrance of the yard next to a small gap in the track and capture the noise of the passing train. From the clicks of wheels, it is possible to identify the type (model) of each locomotive or car the train consists of, allowing administrators to at least keep track of the number of different vehicle types at each site.

Wagon geometry

The *Unified Directory of Wagons*, or in short *UDoW*, is a list of all locomotives and wagons ever used by IGG with a description of the geometry for each of them. The first line of the file, **V**, is the number of vehicles on the list. Each vehicle contains the following lines:

```
L N
W1
W2
...
Wn
```

Where **L** is the length of the car in (integer), **N** is the number of spindles. Each spindle has two wheels mounted; looking from the side, the number of visible wheels matches the number of spindles. The rest **N** lines are spindle (or wheel) offsets from the left buffer (integer). Size units are unknown, but it is guaranteed that the same unit is used for expressing all sizes. For example locomotive 4100 on the illustration would have the following *UDoW* entry:

```
1927 5
505
747
989
1231
1473
```

Each wagon has a unique ID, an integer, determined by the place of the wagon in the *UDoW*: the first entry of *UDoW* is wagon 0, the last described wagon is **V**-1.

Input

Each input is a long sound recording with multiple passing trains and long pauses in between.

Output

Output is a text file, one train per line. Each train is a space separated list of wagon IDs. The submission is accepted only if all trains and all wagons are listed in the order they appeared in the input file, with the proper *UDoW* ID. For example the solution of example.wav is:

```
0 7 7 7 7
0 7 7
```

Fine print

- trains may go from left to right or from right to left
- wagons may be asymmetrical (L-W5 != W1 on the illustration) and occur in both orientations (the locomotive on the illustration may travel from left to right or from right to left)
- each wagon geometry in the *UDoW* is unique (in both orientations)
- the track is straight and the length of the wagon/locomotive from buffer to buffer is exactly the same as specified in the *UDoW*, with 0 extra distance between the buffers of two wagons
- each spindle will cause exactly one loud click at the microphone (but the passing train may produce other clicking sounds at lower volume)
- trains passing at the microphone are travelling at a constant speed that may be different from train to train
- there might be a small amount of noise on timing or in sound (including background noises)
- all inputs were recorded by the same microphone installed on a particular track, and there are no other tracks nearby