НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

Департамент Информатика

Школа „Състезателно програмиране”

СЪСТЕЗАНИЕ, 21 септември 2013 г.

A. The Seven Percent Solution

Uniform Resource Identifiers (or URIs) are strings like *http://icpc.baylor.edu/icpc/*, *mailto:foo@bar.org*, *ftp://127.0.0.1/pub/linux*, or even just *readme.txt* that are used to identify a resource, usually on the Internet or a local computer. Certain characters are reserved within URIs, and if a reserved character is part of an identifier then it must be *percent-encoded* by replacing it with a percent sign followed by two hexadecimal digits representing the ASCII code of the character. A table of seven reserved characters and their encodings is shown below. Your job is to write a program that can percent-encode a string of characters.

Character	Encoding
" " (space)	%20
"!" (exclamation point)	%21
"$" (dollar sign)	%24
"%" (percent sign)	%25
"(" (left parenthesis)	%28
")" (right parenthesis)	%29
"*" (asterisk)	%2a

Input: The input consists of one or more strings, each 1–79 characters long and on a line by itself, followed by a line containing only "#" that signals the end of the input. The character "#" is used only as an end-of-input marker and will not appear anywhere else in the input. A string may contain spaces, but not at the beginning or end of the string, and there will never be two or more consecutive spaces.

Output: For each input string, replace every occurrence of a reserved character in the table above by its percent-encoding, exactly as shown, and output the resulting string on a line by itself. Note that the percent-encoding for an asterisk is %2a (with a lowercase "a") rather than %2A (with an uppercase "A").

Example input:	Example output:
Happy Joy Joy! http://icpc.baylor.edu/icpc/ plain_vanilla (**) ? the 7% solution #	Happy%20Joy%20Joy%21 http://icpc.baylor.edu/icpc/ plain_vanilla %28%2a%2a%29 ? the%207%25%20solution

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

Департамент Информатика

Школа „Състезателно програмиране”

СЪСТЕЗАНИЕ, 21 септември 2013 г.

B. Persistent Bits

WhatNext Software creates sequence generators that they hope will produce fairly random sequences of 16-bit unsigned integers in the range 0–65535. In general a sequence is specified by integers A, B, C, and S, where $1 \le A < 32768$, $0 \le B < 65536$, $2 \le C < 65536$, and $0 \le S < C$. S is the first element (the *seed*) of the sequence, and each later element is generated from the previous element. If X is an element of the sequence, then the next element is (A*X + B) % C where '%' is the remainder or modulus operation. Although every element of the sequence will be a 16-bit unsigned integer less than 65536, the intermediate result A*X + B may be larger, so calculations should be done with a 32-bit *int* rather than a 16-bit *short* to ensure accurate results.

Some values of the parameters produce better sequences than others. The most embarrassing sequences to WhatNext Software are ones that never change one or more bits. A bit that never changes throughout the sequence is *persistent*. Ideally, a sequence will have no persistent bits. Your job is to test a sequence and determine which bits are persistent.

For example, a particularly bad choice is A = 2, B = 5, C = 18, and S = 3. It produces the sequence 3, (2*3+5)%18 = 11, (2*11+5)%18 = 9, (2*9+5)%18 = 5, (2*5+5)%18 = 15, (2*15+5)%18 = 17, then (2*17+5)%18 = 3 again, and we're back at the beginning. So the sequence repeats the same six values over and over:

Decimal	16-Bit Binary
3	0000000000000011
11	0000000000001011
9	0000000000001001
5	0000000000000101
15	0000000000001111
17	0000000000010001
overall	000000000000????1

The last line of the table indicates which bit positions are always 0, always 1, or take on both values in the sequence. Note that 12 of the 16 bits are persistent. (Good random sequences will have no persistent bits, but the converse is not necessarily true. For example, the sequence defined by A = 1, B = 1, C = 64000, and S = 0 has no persistent bits, but it's also not random: it just counts from 0 to 63999 before repeating.) Note that a sequence does not need to return to the seed: with A = 2, B = 0, C = 16, and S = 2, the sequence goes 2, 4, 8, 0, 0, 0, ...

Input: There are from one to sixteen datasets followed by a line containing only 0. Each dataset is a line containing decimal integer values for A, B, C, and S, separated by single blanks.

Output: There is one line of output for each data set, each containing 16 characters, either '1', '0', or '?' for each of the 16 bits in order, with the most significant bit first, with '1' indicating the corresponding bit is always 1, '0' meaning the corresponding bit is always 0, and '?' indicating the bit takes on values of both 0 and 1 in the sequence:

2 5 18 3	000000000000????1
1 1 64000 0	????????????????
2 0 16 2	000000000000???0
256 85 32768 21845	0101010101010101
1 4097 32776 248	0???000011111???
0	

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

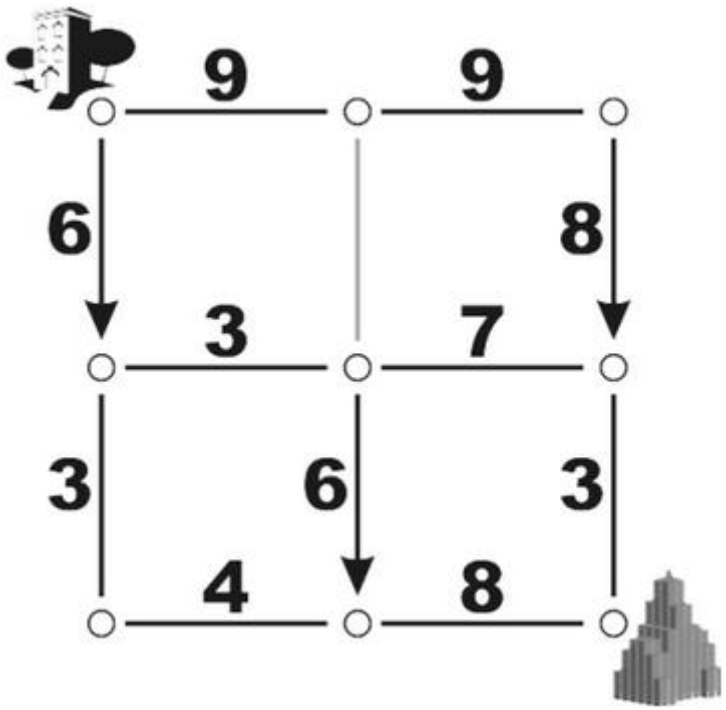
Департамент Информатика
Школа „Състезателно програмиране”
СЪСТЕЗАНИЕ, 21 септември 2013 г.

C. Here We Go(relians) Again

The Gorelians are a warlike race that travel the universe conquering new worlds as a form of recreation. Given their violent, fun-loving nature, keeping their leaders alive is of serious concern. Part of the Gorelian security plan involves changing the traffic patterns of their cities on a daily basis, and routing all Gorelian Government Officials to the Government Building by the fastest possible route.

Fortunately for the Gorelian Minister of Traffic (that would be you), all Gorelian cities are laid out as a rectangular grid of blocks, where each block is a square measuring 2520 rels per side (a rel is the Gorelian Official Unit of Distance). The speed limit between two adjacent intersections is always constant, and may range from 1 to 9 rels per blip (a blip, of course, being the Gorelian Official Unit of Time). Since Gorelians have outlawed decimal numbers as unholy (hey, if you're the dominant force in the known universe, you can outlaw whatever you want), speed limits are always integer values. This explains why Gorelian blocks are precisely 2520 rels in length: 2520 is the least common multiple of the integers 1 through 9. Thus, the time required to travel between two adjacent intersections is always an integer number of blips.

In all Gorelian cities, Government Housing is always at the northwest corner of the city, while the Government Building is always at the southeast corner. Streets between intersections might be one-way or two-way, or possibly even closed for repair (all this tinkering with traffic patterns causes a lot of accidents). Your job, given the details of speed limits, street directions, and street closures for a Gorelian city, is to determine the fastest route from Government Housing to the Government Building. (It is possible, due to street directions and closures, that no route exists, in which case a Gorelian Official Temporary Holiday is declared, and the Gorelian Officials take the day off.)



The picture above shows a Gorelian City marked with speed limits, one way streets, and one closed street. It is assumed that streets are always traveled at the exact posted speed limit, and that turning a corner takes zero time. Under these conditions, you should be able to determine that the fastest route from Government Housing to the Government Building in this city is 1715 blips. And if the next day, the only change is that the closed road is opened to two way traffic at 9 rels per blip, the fastest route becomes 1295 blips. On the other hand, suppose the three one-way streets are switched from southbound to northbound (with the closed road remaining closed). In that case, no route would be possible and the day would be declared a holiday.

Input: The input consists of a set of cities for which you must find a fastest route if one exists. The first line of an input case contains two integers, which are the vertical and horizontal number of city blocks, respectively. The smallest city is a single block, or 1 by 1, and the largest city is 20 by 20 blocks. The remainder of the input specifies speed limits and traffic directions for streets between intersections, one row of street segments at a time. The first line of the input (after the dimensions line) contains the data for the northernmost east-west street segments. The next line contains the data for the northernmost row of north-south street segments. Then the next row of east-west streets, then north-south streets, and so on, until the southernmost row of east-west streets. Speed limits and directions of travel are specified in order from west to east, and each consists of an integer from 0 to 9 indicating speed limit, and a symbol indicating which direction traffic may flow. A zero speed limit means the road is closed. All digits and symbols are delimited by a single space. For east-west streets, the symbol will be an asterisk '*' which indicates travel is allowed in both directions, a less-than symbol '<' which indicates travel is allowed only in an east-to-west direction, or a greater-than symbol '>' which indicates travel is allowed only in a west-to-east direction. For north-south streets, an asterisk again indicates travel is allowed in either direction, a lowercase "vee" character 'v' indicates travel is allowed only in a north-to-south directions, and a caret symbol '^' indicates travel is allowed only in a south-to-north direction. A zero speed, indicating a closed road, is always followed by an asterisk. Input cities continue in this manner until a value of zero is specified for both the vertical and horizontal dimensions.

Output: For each input scenario, output a line specifying the integer number of blips of the shortest route, a space, and then the word "blips". For scenarios which have no route, output a line with the word "Holiday".

Example Input:	Example Output:
2 2 9 * 9 * 6 v 0 * 8 v 3 * 7 * 3 * 6 v 3 * 4 * 8 * 2 2 9 * 9 * 6 v 9 * 8 v 3 * 7 * 3 * 6 v 3 * 4 * 8 * 2 2 9 * 9 * 6 ^ 0 * 8 ^ 3 * 7 * 3 * 6 ^ 3 * 4 * 8 * 0 0	1715 blips 1295 blips Holiday

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

Департамент Информатика
Школа „Състезателно програмиране”
СЪСТЕЗАНИЕ, 21 септември 2013 г.

D. Electronic Document Security

The Tyrell corporation uses a state-of-the-art electronic document system that controls all aspects of document creation, viewing, editing, and distribution. Document security is handled via *access control lists* (ACLs). An ACL defines a set of entities that have access to the document, and for each entity defines the set of rights that it has. Entities are denoted by uppercase letters; an entity might be a single individual or an entire division. Rights are denoted by lowercase letters; examples of rights are *a* for *append*, *d* for *delete*, *e* for *edit*, and *r* for *read*.

The ACL for a document is stored along with that document, but there is also a separate ACL *log* stored on a separate log server. All documents start with an empty ACL, which grants no rights to anyone. Every time the ACL for a document is changed, a new entry is written to the log. An entry is of the form *ExR*, where *E* is a nonempty set of entities, *R* is a nonempty set of rights, and *x* is either "+", "–", or "=". Entry *E*+*R* says to grant all the rights in *R* to all the entities in *E*, entry *E*–*R* says to remove all the rights in *R* from all the entities in *E*, and entry *E*=*R* says that all the entities in *E* have exactly the rights in *R* and no others. An entry might be redundant in the sense that it grants an entity a right it already has and/or denies an entity a right that it doesn't have. A log is simply a list of entries separated by commas, ordered chronologically from oldest to most recent. Entries are cumulative, with newer entries taking precedence over older entries if there is a conflict.

Periodically the Tyrell corporation will run a security check by using the logs to compute the current ACL for each document and then comparing it with the ACL actually stored with the document. A mismatch indicates a security breach. Your job is to write a program that, given an ACL log, computes the current ACL.

Input: The input consists of one or more ACL logs, each 3–79 characters long and on a line by itself, followed by a line containing only "#" that signals the end of the input. Logs will be in the format defined above and will not contain any whitespace.

Output: For each log, output a single line containing the log number (logs are numbered sequentially starting with one), then a colon, then the current ACL in the format shown below. Note that (1) spaces do not appear in the output; (2) entities are listed in alphabetical order; (3) the rights for an entity are listed in alphabetical order; (4) entities with no current rights are not listed (even if they appeared in a log entry), so it's possible that an ACL will be empty; and (5) if two or more consecutive entities have exactly the same rights, those rights are only output once, after the list of entities.

Example input:	Example output:
MC-p,SC+c YB=rde,B-dq,AYM+e GQ+tju,GH-ju,AQ-z,Q=t,QG-t JBL=fwa,H+wf,LD-fz,BJ-a,P=aw #	1:CSc 2:AeBerMeYder 3: 4:BHJfwLPaw

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

Департамент Информатика
Школа „Състезателно програмиране”
СЪСТЕЗАНИЕ, 21 септември 2013 г.

E. Rock Skipping

As a member of the International Rock-Skipping League, you travel to a different lake each week and compete in a rock-skipping contest. The goal is to throw a rock so that it skips as many times as possible; the exact rules for determining the winner are given below. To make the competitions interesting, the IRSL often chooses lakes with logs, sandbars, and other obstacles. You are provided with a side-view, water-level "map" of the lake as shown in the top line of the example below. (The numbers 0..29 below the map are just for reference.) A period (".") indicates clear water, where a rock will skip; any other character indicates some kind of obstacle that will stop a rock.

```
...=...**..#...@....:.:.:..=..
_____________________________
            11111111112222222222
012345678901234567890123456789
```

You stand at the left end of the lake. You can throw a rock so that it lands at any position in the lake, and then skips at any fixed interval thereafter. So a throw can be defined as a pair (i,d), where $i \geq 0$ is the initial landing position and $d > 0$ is the distance between skips. Note that d must be positive. The *count* of a throw is the number of times that it skips on the water. The *length* is the position of its last contact with either the water or an obstacle. To rank two distinct throws, use the following criteria, in order, until a winner is determined: count (highest wins); length (greatest wins); initial position (greatest wins); distance between skips (smallest wins).

For the map shown above, throw (27,2) hits the obstacle at position 27; it has count 0 and length 27. Throw (16,1) skips at positions 16, 17, 18, and 19, then hits the obstacle at position 20; it has count 4 and length 20, so it beats throw (27,2). Throw (2,7) skips at positions 2, 9, 16, and 23, then skips over the lake; it has count 4 and length 23, so it beats throw (16,1). Throw (1,4) skips at positions 1, 5, 9, 13, 17, 21, 25, and 29, then skips over the lake; it has count 8 and distance 29, and is the best possible throw for this lake.

Input: The input consists of one or more lake maps, each 1–40 characters long and on a line by itself, followed by a line containing only "END" that signals the end of the input. Positions within a map are numbered starting with zero. Maps will only contain printable ASCII punctuation characters. A period indicates clear water and any other character indicates an obstacle.

Output: For each map, compute the best possible throw (i,d), then output a line containing i and d separated by one space.

There is guarantee that all judge's tests has only one result!

Example input:	Example output:
...=...**..#...@....:.:.:..=.. .(+) /^\. *++&* END	1 4 0 3 3 1 4 1

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

Департамент Информатика
Школа „Състезателно програмиране”
СЪСТЕЗАНИЕ, 21 септември 2013 г.

F. Misspelling

Misspelling is an art form that students seem to excel at. Write a program that removes the *n*th character from an input string.

Input

The first line of input contains a single integer ***N***, ($1 \le N \le 1000$) which is the number of datasets that follow.

Each dataset consists of a single line of input containing ***M***, a space, and a single word made up of uppercase letters only**.** ***M*** will be less than or equal to the length of the word. The length of the word is guaranteed to be less than or equal to 80.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the misspelled word. The misspelled word is the input word with the indicated character deleted.

Sample Input	Sample Output
4 4 MISSPELL 1 PROGRAMMING 7 CONTEST 3 BALLOON	1 MISPELL 2 ROGRAMMING 3 CONTES 4 BALOON

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

Департамент Информатика
Школа „Състезателно програмиране”
СЪСТЕЗАНИЕ, 21 септември 2013 г.

G. Conversions

Conversion between the *metric* and *English* measurement systems is relatively simple. Often, it involves either multiplying or dividing by a constant. You must write a program that converts between the following units:

Type	Metric	English equivalent
Weight	1.000 kilograms	2.2046 pounds
	0.4536 kilograms	1.0000 pound
Volume	1.0000 liter	0.2642 gallons
	3.7854 liters	1.0000 gallon

Input

The first line of input contains a single integer ***N***, ($1 \le N \le 1000$) which is the number of datasets that follow.

Each dataset consists of a single line of input containing a floating point (double precision) number, a space and the *unit specification* for the measurement to be converted. The *unit specification* is one of **kg**, **lb**, **l**, or **g** referring to kilograms, pounds, liters and gallons respectively.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the appropriately converted value rounded to 4 decimal places, a space and the *unit specification* for the converted value.

Sample Input	Sample Output
5 1 kg 2 l 7 lb 3.5 g 0 l	1 2.2046 lb 2 0.5284 g 3 3.1752 kg 4 13.2489 l 5 0.0000 g

НБУ
НОВ БЪЛГАРСКИ УНИВЕРСИТЕТ
София 1635, ул. Монтевидео 21
тел.: 55 81 37, 55 21 35, факс: 957 19 30

Департамент Информатика
Школа „Състезателно програмиране”
СЪСТЕЗАНИЕ, 21 септември 2013 г.

H. Filthy Rich

They say that in Phrygia, the streets are paved with gold. You're currently on vacation in Phrygia, and to your astonishment you discover that this is to be taken literally: small heaps of gold are distributed throughout the city. On a certain day, the Phrygians even allow all the tourists to collect as much gold as they can in a limited rectangular area. As it happens, this day is tomorrow, and you decide to become filthy rich on this day. All the other tourists decided the same however, so it's going to get crowded. Thus, you only have one chance to cross the field. What is the best way to do so?

Problem

Given a rectangular map and amounts of gold on every field, determine the maximum amount of gold you can collect when starting in the upper left corner of the map and moving to the adjacent field in the east, south, or south-east in each step, until you end up in the lower right corner.

Input

The input starts with a line containing a single integer, the number of test cases.

Each test case starts with a line, containing the two integers r and c, separated by a space ($1 \leq r; c \leq 1000$). This line is followed by r rows, each containing c many integers, separated by a space. These integers tell you how much gold is on each field. The amount of gold never negative.

The maximum amount of gold will always fit in an int.

Output

For each test case, write a line containing "Scenario #i:", where i is the number of the test case, followed by a line containing the maximum amount of gold you can collect in this test case. Finish each test case with an empty line.

Sample Input	Sample Output
1 3 4 1 10 8 8 0 0 1 8 0 27 0 4	Scenario #1: 42