16^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

4 October 2014

Problem A — Blue Balloon Amusement Park

Problem Description

An amusement park is planning a special offer for their opening day. At the exit from every ride, there is an arrow pointing at some ride (possibly the same one). The first person to arrive at each ride after the gates open will enjoy free rides for the day, but only as long as she follows the arrows from one ride to the next.

The owners want to determine how many free rides they will give away. They assume that a customer will take the free rides as long as each ride is new: once the arrow points to a ride she has already taken, she will get bored and leave. Also, a customer cannot be first at more than one ride.

Input

The input consists of an arbitrary number of records, but no more than 20. Each record starts with a line containing an integer N, the number of rides, which are numbered from 1 to N. This is followed by a line containing N integers, a_1 to a_N, separated by spaces. The arrow at the exit from ride i points to ride a_i. The end of input is marked by a line containing only the value -1.

In all test cases, $1 \leq N \leq 100\,000$.

Output

For each test case, output a line containing the total number of free rides that will be given away. Note that the answer might not fit in a 32-bit integer.

Sample input

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

Sample output

```
14
```

Time limit

5 seconds

16^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

4 October 2014

Problem B — Yellow Balloon
Lucky Digit

Problem Description

John believes that the digit D is lucky, and looks out for it in numbers everywhere. When he learned how to represent numbers of bases other than 10, he was very excited, because some numbers will be luckier in other bases. He wants your help to determine, for each number he gives you, the luckiest representation in any base from 2 to 10, inclusive.

For example, suppose $D = 7$, and John gives you the number 507 (in base 10). In base 8 this would be written 773 (because $7 \cdot 8^2 + 7 \cdot 8^1 + 3 \cdot 8^0 = 507$), and in base 9 it would be written 623. Since 773 has the most 7s in it, this is considered the luckiest representation. If there is a tie, the representation in the higher base is considered luckier.

Input

The input consists of an arbitrary number of records, but no more than 50. Each record is a line containing two integers (in base 10), N and D, separated by a space. The end of input is marked by a line containing only the value -1.

In all test cases, $0 \leq N \leq 1\,000\,000$ and $0 \leq D \leq 9$.

Output

For each test case, output a line containing the luckiest representation of N for the lucky digit D.

Sample input

```
507 7
64 0
123 9
-1
```

Sample output

```
773
1000000
123
```

Time limit

5 seconds

16^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

4 October 2014

Problem C — Red Balloon
Your move!

Problem Description

You are building a chess computer to give old Garry another thrashing. But first things first, which means writing an algorithm to check whether a particular chess move is valid.

Chess is played on a 8×8 chequered board using six different pieces: pawns, rooks, bishops, knights, kings and queens. Examples of the valid moves are shown in Figure 1. Black dots indicate valid positions that a particular piece may move to.

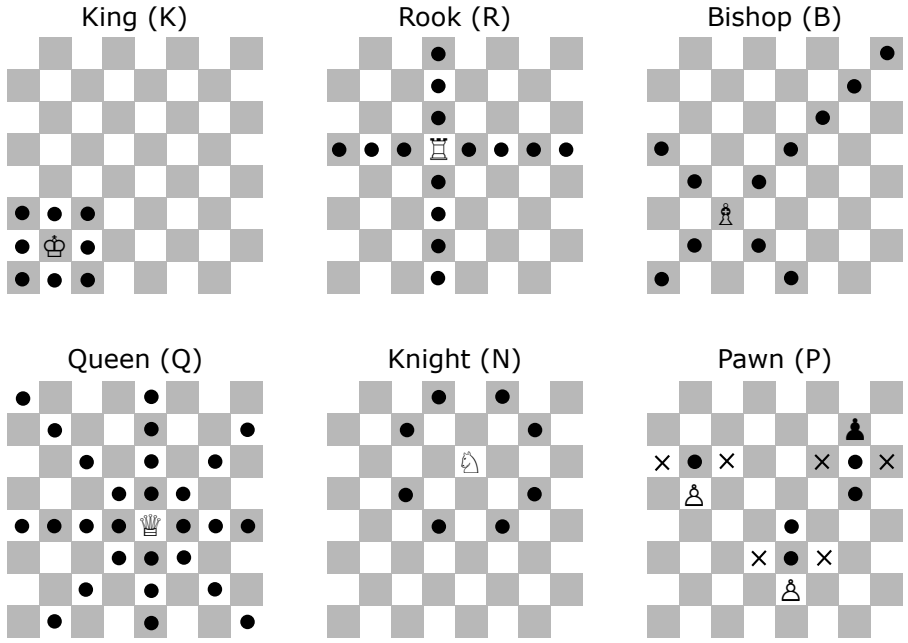


Figure 1: Chess piece moves

Some general rules apply to the movements as shown in Figure 1: No piece is allowed to "move through" any other piece. A move may end in a square that is occupied by an opponent's piece, which results in the "capturing" of the opponent's piece; moving onto a square occupied by a piece of the same colour is not allowed.

The moves of the pieces may be defined as follows, subject to the general rules stated so far (adapted from the FIDE "Laws of chess"):

- A bishop may move to any square along a diagonal on which it stands;
- A rook may move to any square along the rank (row) or file (column) on which it stands;
- A queen may move to any square along the rank, file or diagonal on which it stands;

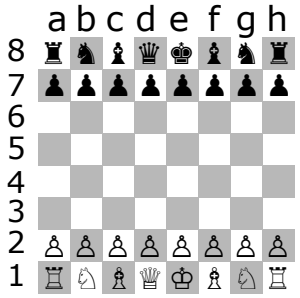


Figure 2: Chess board coordinates and starting positions. Rows are called 'ranks', and columns are called 'files'.

- A knight may move two ranks and one file away from the square on which it stands, or one rank and two files. The knight is an exception in that it "jumps over" the squares in between its current position and the destination square, so that the contents of these squares do not impact on the validity of a move (subject to the constraints of 'check', defined below).
- A king may move to any adjoining square;
- A pawn may move forward to the unoccupied square immediately in front of it on the same file — for White, this means to higher-numbered ranks, and for Black, lower numbered ranks (see Figure 2). A white pawn in rank 2 or a black pawn in rank 7 is allowed to advance by two squares provided both squares are unoccupied. A pawn may move to a square occupied by an opponent's piece diagonally in front of it on an adjacent file, capturing it in the process (marked $\times$ in Figure 1).

In addition to the movement rules above, valid moves have to respect the concept of being 'in check':

> The king is said to be 'in check' if it is attacked by one or more of the opponent's pieces, even if such pieces are constrained from moving to that square because they would then leave or place their own king in check. No piece can be moved that will either expose the king of the same colour to check or leave that king in check.

You do not have to account for advanced moves like *en passant*, castling or tilting.

Given a chess board configuration, you must determine whether a proposed move is valid, or not. Using the numbering scheme depicted in Figure 2, a proposed move can unambiguously be described by providing only the starting and ending coordinates.

Input

Your input consists of an arbitrary number of records, but no more than 100. Each record starts with an integer n, with $2 \leq n \leq 32$, denoting the number of piece placement definitions to follow. Each placement definition takes the form

$$[\mathrm{W}]X\ yz$$

where the optional prefix 'W' denotes the piece is white (as opposed to black), and $X \in \{\mathrm{K}, \mathrm{R}, \mathrm{B}, \mathrm{Q}, \mathrm{N}, \mathrm{P}\}$ denotes the piece type as shown in Figure 1; there is no space between the optional 'W' and the piece type denoted by X. The coordinate yz is composed of the rank $y \in \{1, 2, 3, 4, 5, 6, 7, 8\}$ followed by the file $z \in \{\mathrm{a}, \mathrm{b}, \mathrm{c}, \mathrm{d}, \mathrm{e}, \mathrm{f}, \mathrm{g}, \mathrm{h}\}$ without an intervening space. Figure 2 illustrates the coordinate system.

After the n placement definitions, a single move command of the form

$$xy\ uv$$

follows indicating that the piece located at rank/file xy must be moved to rank/file uv, using the same rank/file notation as in the placement definitions. For all input records, it may be assumed that $xy \neq uv$, that there is a piece at xy, that no two pieces will occupy the same square, each player will have exactly one king, and at most one player will be in check.

Strictly speaking, in chess White always moves first, but for the purpose of this problem it is not considered an invalid move if it happens that Black makes the first move.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output

```
valid
```

or

```
invalid
```

to indicate whether the move proposed in the record is valid or invalid according to the chess rules defined above.

Sample input

```
4
K 8a
R 7b
WB 3f
WK 1b
7b 1b
4
K 8a
R 7b
WB 3f
WK 1b
1b 1a
5
K 8a
R 7b
WB 3f
WK 1b
WP 4h
4h 6h
5
K 8a
R 7b
WB 3f
WK 1b
P 6h
6h 5h
-1
```

Sample output

```
invalid
valid
invalid
valid
```

Time limit

1 second

16^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

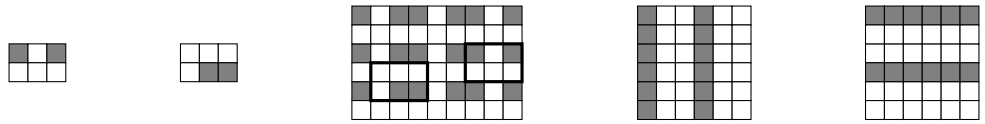
4 October 2014

Problem D — Orange Balloon Floor Tiles

Problem Description

You have been hired to design floor tiles for a new building. A room will be tiled with copies of a template tile, where each template is a rectangle made from $M \times N$ solid-colour squares. There are C different colours available for the squares.

The architects want to know how many possible floor patterns there are. Note that two different-looking tiles can produce the same pattern, just with a shift. For example, the tiles marked as (a) and (b) in the figure below both lead to the pattern (c). This pattern should only be counted once. However, if one pattern must be rotated to match another, they are considered different, such as (d) and (e).



(a) (b) (c) (d) (e)

As an example, here are the tiles that generate the unique patterns for $M = 2, N = 3, C = 2$:

Input

The input consists of an arbitrary number of records, but no more than 20. Each record is a line containing three integers M, N and C, separated by spaces. The end of input is marked by a line containing only the value -1.

In all test cases, $1 \leq M, N \leq 100$ and $1 \leq C \leq 1000$.

Output

For each test case, output a single line containing the number of distinct tiles, modulo $1\,000\,000\,007$.

Sample input

```
1 2 3
1 4 2
3 2 2
10 12 17
-1
```

Sample output

```
6
6
14
732859706
```

Time limit

5 seconds

16^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

4 October 2014

Problem E — Purple Balloon Median

Problem Description

A common tool in statistics is the *median*. Given a list of numbers with an odd length, the median is the one that would appear in the middle when they are placed in increasing order. For example, consider the list 20, 14, 4, 10, 10. In sorted order, this would be 4, 10, 10, 14, 20, and the middle element is 10. When the length of the list is even, there will be two numbers that are in the middle: the median is the mean (half the sum) of these numbers.

Given a list of numbers, all of them even, compute their median.

Input

The input consists of an arbitrary number of records, but no more than 20. Each record starts with a line containing an integer N, the length of the list. The next line contains N even integers, separated by spaces. The end of input is marked by a line containing only the value -1.

In all test cases, $1 \leq N \leq 10$, and $0 \leq a_i \leq 100$ for each number a_i in the list.

Output

For each test case, output a line containing the median of the list.

Sample input

```
5
20 14 4 10 10
4
20 14 4 10
-1
```

Sample output

```
10
12
```

Time limit

5 seconds

16^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

4 October 2014

Problem F — Green Balloon
Gopher trouble

Problem Description

Your garden has been infested by cybernetically enhanced gophers — just one of the many joys of living close to an Umbrella Corporation research lab. As gophers are wont to do, they have dug numerous holes all across your lawn.

Fortunately, you were prepared for this scenario. You have a fully automated mortar launcher, combined with coordinates of the locations of the gopher holes (obtained with your aerial drone, of course). The mortar launcher is controlled through three parameters: elevation angle, azimuth angle, and muzzle velocity. Figure 1 depicts the role of the elevation and azimuth angles: elevation angle is the angle of the launcher barrel relative to the ground plane, and azimuth angle defines the direction in which the barrel is pointed within the ground plane, as seen from overhead.

The planar trajectory that the mortar follows is defined by the linear air resistance equations (note: the notation $\vec{V}$ indicates that V is a vector):

$$\vec{P}(t) = \vec{V}_1 \frac{m}{k} e^{k \cdot t/m} - \vec{G} \frac{m \cdot g \cdot t}{k} + \vec{P}_1$$

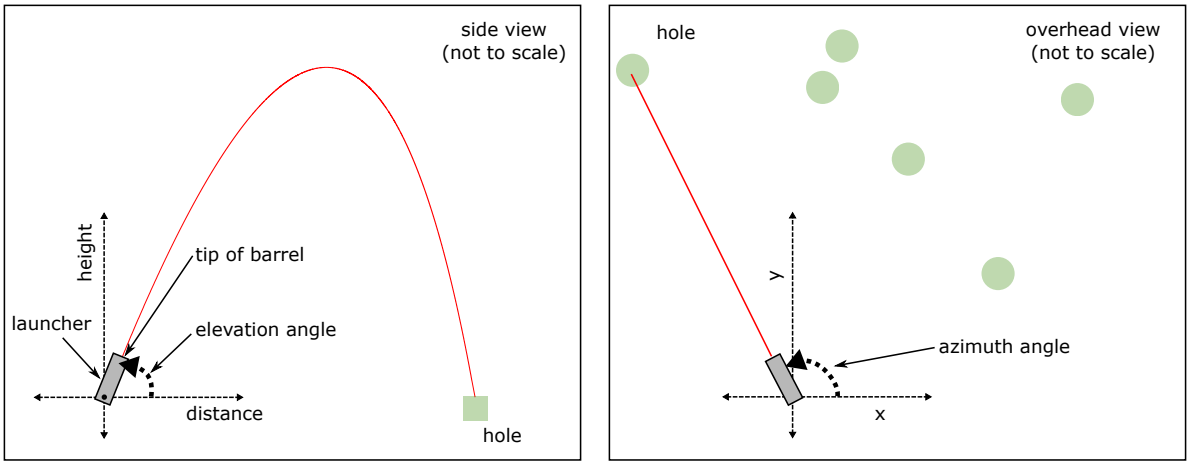


Figure 1: Trajectory

where (note: square brackets [ ] are used to group vector components)

$\vec{P}(t)$	=	a vector indicating the position of object relative to its position at time 0
t	=	time in seconds; $t = 0$ when projectile leaves the muzzle
$\vec{V}_0$	=	[horizontal muzzle velocity, vertical muzzle velocity]
$\vec{V}_1$	=	$\vec{V}_0 + \vec{G}\frac{m \cdot g}{k}$
$\vec{G}$	=	$[0, -1]$. This is the direction of gravity
m	=	mass of mortar, fixed at 0.145 kg
g	=	9.81 m·s^{-2}
k	=	viscosity coefficient, fixed at -0.05 kg·s^{-1}
$\vec{P}_0$	=	[initial horizontal position, initial vertical
$\vec{P}_1$	=	$\vec{P}_0 - \vec{V}_1\frac{m}{k}$
position]		
l	=	barrel length, fixed at 0.4 m

The initial conditions $\vec{P}_0$ and $\vec{V}_0$ depend upon the elevation angle (provided as an input); in addition, $\vec{P}_0$ further depends on the barrel length, l (which is defined as 0.4 m). The pivot point of the launcher is defined to be the origin, i.e., the vector $[0, 0, 0]$. The muzzle velocity is defined as the velocity of the mortar as it leaves the barrel.

To ensure that the mortar clears the entrance to the gopher hole, the mortar must remain clear of the sides of the hole down to a depth of 0.03 m. Imagine a vertical line running through the centre of the hole into the ground, indicated as the "hole centreline" in Figure 2. A shot is said to be a successful "hit" if the centre of the mortar remains within a distance of 0.005 m of this vertical line between ground level (depth of 0 m) and a depth of 0.03 m below ground level, and is declared a "miss" otherwise.

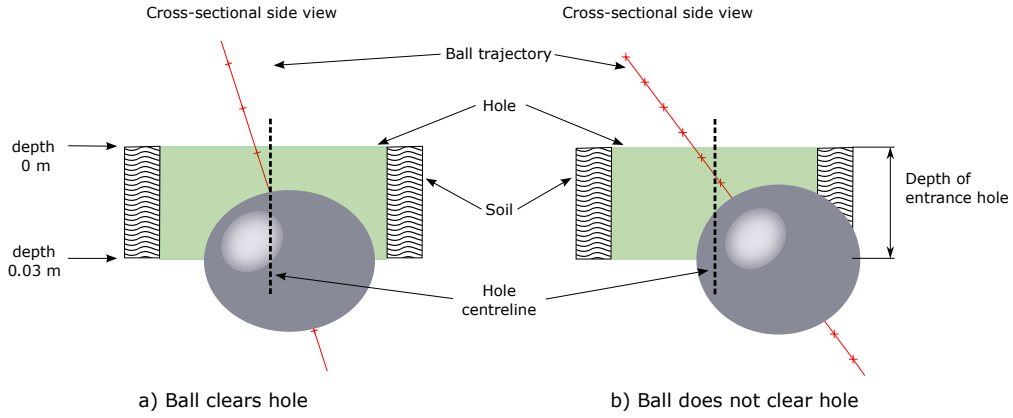


Figure 2: Does the ball clear the hole?

Given an elevation angle, an azimuth angle, and a pair of $[x, y]$ target coordinates, you must evaluate whether it is possible adjust the launcher's muzzle velocity to score a successful hit on the target.

Input

Your input consists of an arbitrary number of records, but no more than 2000. Each record comprises four space-separated values e a x y. The values e and a denote the elevation and azimuth angles, and are specified in degrees. The values x and y denote the centre location of the target hole, as seen from the overhead view in Figure 1, in metres.

The following constraints apply: $20 \leq e \leq 88$; $5 \leq a \leq 175$; $-100 \leq x \leq 100$; $0 \leq y \leq 100$; $8 \leq \sqrt{x^2 + y^2} \leq 100$.

Given these input parameters, you must determine whether the mortar launcher muzzle velocity (constrained to the range [1,500] $\mathrm{m \cdot s^{-1}}$) can be adjusted to score a hit.

Internal calculations should be performed using double-precision floating point number representations, with a recommended tolerance of 10^{-10}. All inputs are guaranteed to have a well-defined solution that is not affected by minor truncation errors.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output

```
hit
```

or

```
miss
```

as appropriate, where "miss" is output only if no solution could be found that would allow the mortar to clear the hole as defined above.

Sample input

```
68.23 9.28 76.075364 12.430553
33.47 156.44 -54.295026 23.675787
56.78 129.66 -58.640789 73.300483
-1
```

Sample output

```
hit
miss
miss
```

Time limit

5 seconds

16^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

4 October 2014

Problem G — White Balloon
Angles

Problem Description

Some people claim that Stonehenge and other stone circles were built by aliens. They point out that when one selects the right circles and plots them on a map, precise geometric shapes appear. However, given enough points to choose from, it will be easy to find geometric shapes.

As a demonstration, you must find right angles in given sets of points, which could represent anything — stone circles, ancient burial grounds, pyramids, or even caravan parks. Specifically, out of the given points, you must find the triplet (A, B, C) of distinct points such that the angle between AB and BC is as close to 90° as possible.

Input

The input consists of an arbitrary number of records, but no more than 20. Each record starts with a line containing N, the number of points. This is followed by N lines, each containing two integers x y separated by a space, the coordinates of one of the points. The end of input is marked by a line containing only the value -1.

In all test cases, $3 \leq N \leq 1000$ and $-2000 \leq x, y \leq 2000$. No two points in a test case will have the same coordinates, and the points in a test case will not all lie on the same line.

Output

For each test case, output a line of the form

```
a b c
```

where a, b, c are the indices of the points A, B and C. Points are numbered from 1 to N in the order they appear in the input. Break ties by choosing the solution with the smallest value for a; if there are still ties, choose the smallest value for b, and finally the smallest value for c.

Sample input

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

Sample output

```
1 2 4
2 4 3
1 5 6
1 2 6
```

Time limit

10 seconds

16^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

4 October 2014

Problem H — Pink Balloon
Circular reasoning

Problem Description

You are conducting a study into the painting habits of a rare kind of urban graffiti artist. The *Ensō* (as they call themselves) paint only a few circles in a rectangular empty space, with a tendency to allow the circles to overlap. To appreciate the state of mind of the artist, you are reconstructing the ways in which a particular painting could have been painted. Specifically, given a painting, you are counting the possible orderings in which the set of overlapping circles could have been drawn.

A graffiti artwork is represented as a rectangular grid of symbols. In this representation, you have identified only three sizes of circles (radius of 3, 4 or 5), as shown below:

```
-------------    -------------    -------------
-------------    -------------    ----AAAAA----
-------------    -----CCC-----    ---A-----A---
-----BBB-----    ----C---C----    --A-------A--
----B---B----    ---C-----C---    -A---------A-
---B-----B---    --C-------C--    -A---------A-
---B-----B---    --C-------C--    -A---------A-
---B-----B---    --C-------C--    -A---------A-
----B---B----    ---C-----C---    -A---------A-
-----BBB-----    ----C---C----    --A-------A--
-------------    -----CCC-----    ---A-----A---
-------------    -------------    ----AAAAA----
-------------    -------------    -------------
```

Note that empty space (no paint) is indicated with a minus ("-") symbol, and that the colour used to paint the circle is encoded with an uppercase letter in the range A to I (inclusive).

A sample artwork, composed of two circles, might look like this:

```
--BBB--
-B-AAB-
B-A---B
BA----B
BA----B
-B---B-
--BBB-A
```

The circle in colour 'B' clearly overdraws the circle painted in colour 'A'; thus there is only one possible drawing order that produces this particular artwork. Note that circles may be truncated by the edges of the drawing area.

Input

Your input consists of an arbitrary number of records, but no more than 20. Each record starts with a pair of space-separated integers r c, denoting the number of rows (r) and columns (c) in the artwork, with $7 \leq r, c \leq 25$. The next r rows each consist of c characters from the set {A–I,-}.

Each input record is guaranteed to have a minimum of three visible symbols for each circle found in that drawing. A given colour (if present) is used in only one circle per drawing.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output

`k`

where `k` denotes the number of possible orderings in which the circles in the given painting could have been drawn.

Sample input

```
15 15
---CCC----EEE--
--CBB-C--E---E-
-C---B-CE-----E
BC----BCE-----E
BC----BCE-----E
B-C---C-AEF--E-
-B-CCC-A--EEE--
--BBB-A-----F--
---DDD-------F-
--D--FD------F-
-D---F-D-----F-
-D----FD----F--
-D-----D---F---
--D---D-FFF----
---DDD---------
11 11
---C-------
---CAAA----
--CABBBA---
CCAB---BA--
-AB-----BA-
-AB-----DDD
-AB----DBA-
--AB--DBA--
---ABDBA--E
----ADA--E-
-----D--E--
-1
```

Sample output

```
12
40
```

Time limit

10 seconds