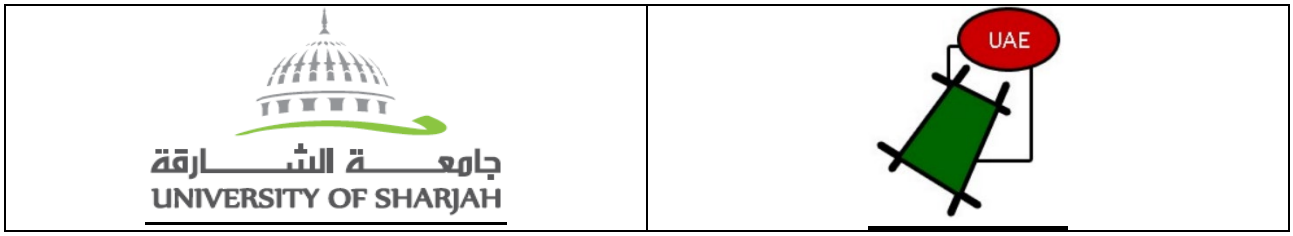


University Programming Contest
UPC 2022 – 2023
March 9th, 2023

University of Sharjah, UAE
Problem Set
Duration: 10:00 – 14:00

Problem #	*Problem Name*	*Balloon Color*
A	Coin Box	Purple
B	Neighbours	Red
C	Maximum Distance	Blue
D	Clock Angle	White
E	Binary Weight	Black
F	Shalaby's Cipher	Green
G	Formatting	Pink
H	Super Knight	Gold
I	Casting Budget	Yellow

A. Coin Box

Program: coin. (cpp | java | py)

Description

A couple created a small contest between them, the winner will decide where to go next weekend. They will continuously and alternately draw a coin from a coin box, they will sum them up until one of them (the winner) reaches 100 cents. Inside the coin box there are 10, 20, 50 cents coins. The box contains enough coins of each type. Your task is to insert the inputs alternately for the husband and wife and sum them up until one of them reaches 100 cents, then the program should print who is the winner.

Input

The first line of input starts with an integer N denoting the number of test cases.

For every test case, each line contains X and Y, separated by a space. X is the input of the husband and Y is the input of the wife. The last line of each case contains-1 and-1, indicating the end of the case.

Output

For each case, output one sentence stating the winner.

Sample Input / Output

Input
3 10 20 50 20 10 50 10 20 -1-1 50 50 20 10 10 20 20 10 -1-1 50 50 10 50 -1-1

Output
The winner is the wife. The winner is the husband. The winner is the wife.

B. Neighbours

Program: neighbours. (cpp | java | py)

Description

The purpose of this problem is to develop an application that displays for any element in a two-dimensional array of 5 rows and 5 columns its north, south, west, and east neighbours. The elements of the array are provided as described in the input. If an element has no neighbours to a particular direction (e.g. North) the program must print “null” as the neighbour. The position of the element for which the neighbours have to be determined is also read as the last input.

Example

	0	1	2	3	4
0	0	1	2	3	4
1	5	6	7	8	0
2	1	2	3	4	5
3	6	7	8	0	1
4	2	3	4	5	6

Element Position (1, 1): 6
North neighbours: 1
South neighbours: 2, 7, 3
West neighbours: 5
East neighbours: 7, 8, 0

Input

In the first 5 lines, there are 5 integers selected from the list {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.
On the sixth line, an integer N is given representing the number of test cases. On each of the next N lines, there are two integers representing the row number and the column number.

Output

For each test case, the program prints 4 lines listing the neighbouring integers from west, east, north, and south.

Sample Input / Output

Input

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

Output

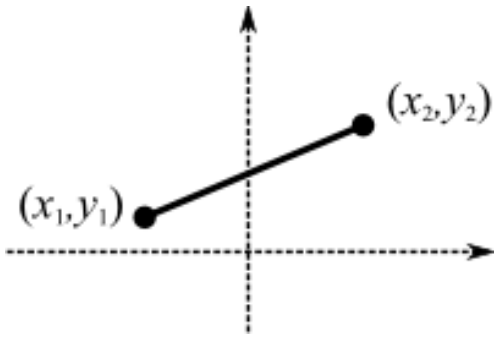
```
The west neighbours of element 4 at position [2][0] are: null end of list
The east neighbours of element 4 at position [2][0] are: 7 8 5 1 end of list
The north neighbours of element 4 at position [2][0] are: 0 9 end of list
The south neighbours of element 4 at position [2][0] are: 2 1 end of list
The west neighbours of element 4 at position [3][2] are: 2 3 end of list
The east neighbours of element 4 at position [3][2] are: 2 9 end of list
The north neighbours of element 4 at position [3][2] are: 2 7 8 end of list
The south neighbours of element 4 at position [3][2] are: 5 end of list
```

C. Maximum Distance

Program: distance. (cpp | java | py)

Description

You are provided with N pairs of points: **(x1, y1)** and **(x2, y2)**. You need to find the pair with the maximum distance.



Input

N (depicting the number of pair of points that will follow)

N pairs of points on each line: Each line will have four integers x1, y1, x2 and y2.

The input will end with-1.

Output

The output will be the pair with the maximum distance and the distance between two points.

Sample Input / Output

Input
3 4 5 10 2 3 1 0 1 6 9 100 4 2 8 9-10 2 20 7 5 3 -1

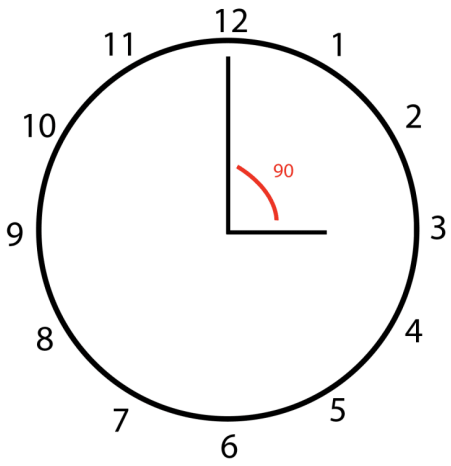
Output
6 9 100 4 94.1329 8 9-10 2 19.3132

D. Clock Angle

Program: clock. (cpp | java | py)

Description

Find the angle between the minutes hand and the hours hand on a regular analogue clock. Assume that the seconds hand, if there were one, would be pointing straight up at 12. Give all angles as the smallest positive angles. For example, 9:00 is 90 degrees; not -90 or 270 degrees.



Input

The first line specifies the number of test cases. The input is a list of times in the form 'H M', each on their own line, with $1 \leq H \leq 12$ and $00 \leq M \leq 59$. Note that H may be represented with 1 or 2 digits (for 1–9 or 10–12, respectively); M is always represented with 2 digits (the input times are what you typically see on a digital clock).

Output

The output displays the smallest positive angle in degrees between the hands for each time. The answer should between 0 degrees and 180 degrees for all input times. Display each angle on a line by itself in the same order as the input. The output should be rounded to the nearest 1/1000, i.e., three places after the decimal point should be printed.

Sample Input / Output

Input
3 12 00 9 00 8 10

Output
0.000 90.000 175.000

E. Binary Weight

Program: binary. (cpp | java | py)

Description

The binary weight of a number is the amount of 1s in the number's binary representation.
For example, 43 in binary is 101011, so the binary weight is 4.
Given a decimal number, we want to find the next greater decimal number that has the same binary weight. In this case, 45 (101101) is such a number.

Input

The first line of input will contain T (≤ 100) denoting the number of cases.
Each test case consists of a single line containing a number x ($0 < x < 1000000$).

Output

Print the answer for each test case (corresponding to the next decimal number with the same binary weight as in the input) on a separate line.
Reminder: a binary representation of a number is the sum of powers of 2, where 1 means that power is included, and 0 means that it's not. So, a 101011 (binary)

$= 1*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$
$= 1*32 + 0*16 + 1*8 + 0*4 + 1*2 + 1*1$
$= 32 + 8 + 2 + 1$
$= 43$ (decimal)

Sample Input / Output

Input
4 3 4 10 43

Output
5 8 12 45

F. Shalaby's Cipher

Program: cipher. (cpp | java | py)

Description

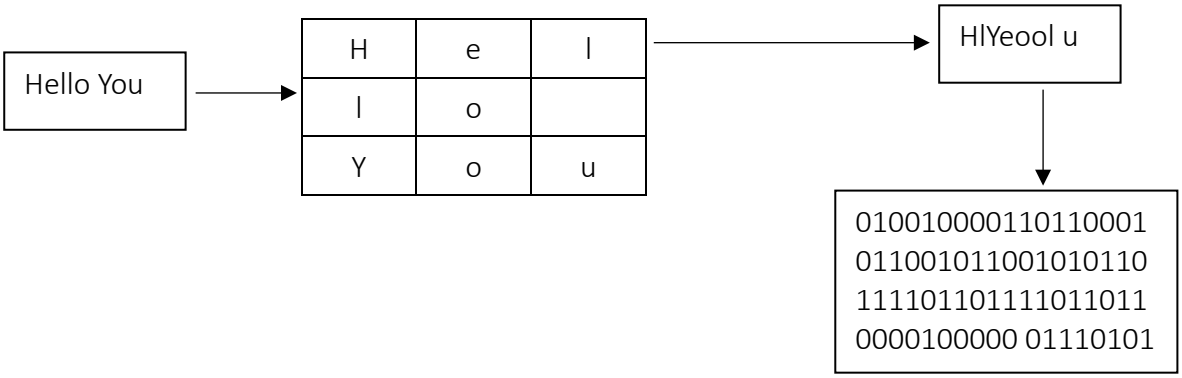
Shalaby just finished reading a mystery book that relied on cryptology and his mind is completely blown by classical era ciphers. Since then, Shalaby has spent countless hours running down several internet rabbit holes reading about them.

As a College of Computing and Informatics student, Shalaby was already enamoured by the concept of binary numbers and loved to doodle during lectures by converting text from character to binary and vice versa using the ASCII code.

So, when Shalaby read about the scytale cipher, a cipher that was used during classical eras he decided to use it for his own personal cipher.

Whenever he wants to send his friends a message: First, he encrypts the message using the scytale cipher, and then he converts the encrypted message to binary.

To encrypt using the scytale cipher, Shalaby writes his message in a nxn grid, by filling each row and starting the next, (each cell contains only one character), where n is the square root of the length of the message (Obviously, he only writes messages with perfect squares lengths). Then, he writes the encrypted message by copying the characters across columns (See figure below). As a last step, Shalaby encodes the characters to their binary representation using ASCII (each character is represented with 8 bits).



As Shalaby's friend, you need to write a program to decode from binary to ASCII code and then use the scytale cipher to decrypt the message.

Input

The first line contains k, indicating the number of cases, where $0 < k < 10$,
For k number of times, your program should read a string of binary bits no longer than 8000 characters, and convert it to the decrypted message, by decoding it into ASCII characters, and decrypting it through the scytale cipher.

Output

Your program should output k decrypted messages.

Sample Input / Output

Input	Output
2 01000010011100110110010101110100 01001000011110010110010100100001	Best Hey!

G. Formatting

Program: format. (cpp | java | py)

Conditional Formatting allows the variation of the presentation depending on certain properties of the information. For example, when presenting a bank statement, the bank may decide to display in red any transaction with an amount above AED 5,000 in order to grab the attention of the user.

Another technique, frequently used in printing tables, is to alternate the background color of rows to make it easier for the reader to visually follow a row. For example, the background color in Table 1 alternates every three rows, while in Table 2, the color alternates after each row.

Character	Decimal	Binary
A	65	01000001
B	66	01000010
C	67	01000011
D	68	01000100
E	69	01000101
F	70	01000111
A	97	01100001
b	98	01100010
c	99	01100011
d	100	01100100
e	101	01100101
f	102	01100110

Table 1

Name	Unit
Byte	B
Kilobyte	KB
Megabyte	MB
Gigabyte	GB
Terabyte	TB
Petabyte	PB

Table 2

A properly designed template language would have a construct to allow the designer to alternate the properties of table rows. In this problem, we shall concentrate on one such construct that takes three arguments: *N*, *P*1, and *P*2. The template engine would then apply *P*1 on the first *N* rows, *P*2 on the second *N* rows, and then back to *P*1 on the third *N* rows, and so on.

Write a program that takes the current row number (starting at one,) the number ***N***, and properties ***P*1**, and ***P*2** and determines which of *P*1 or *P*2 should be applied to the current row.

Input

Your program will be tested on one or more test cases. Each test case is specified on a separate line. Each line specifies four values: **R**, **N**, **P1**, and **P2**, all separated by one or more spaces.
R is the current row number (first row is numbered 1) while **N** is as described above.
Note that 0 < **R,N** < 1,000.
P1 and **P2** are properties. A property is a string made of upper- or lower-case letters, digits, and/or spaces. A property may be surrounded by double quotes, (but the double quotes are not part of the property.) If a property contains spaces, the surrounding double quotes are mandatory.
No property will be longer than **512** characters (including the double quotes if present.)
The last line of the input is made of a single zero.

Output

For each test case, output the result on a single line using the following format:
k. result

Where k is the test case number (starting at 1,) and result is P1 or P2.
Note that the double quotes are never printed. In addition, all letters are printed in lower case.

Sample Input / Output

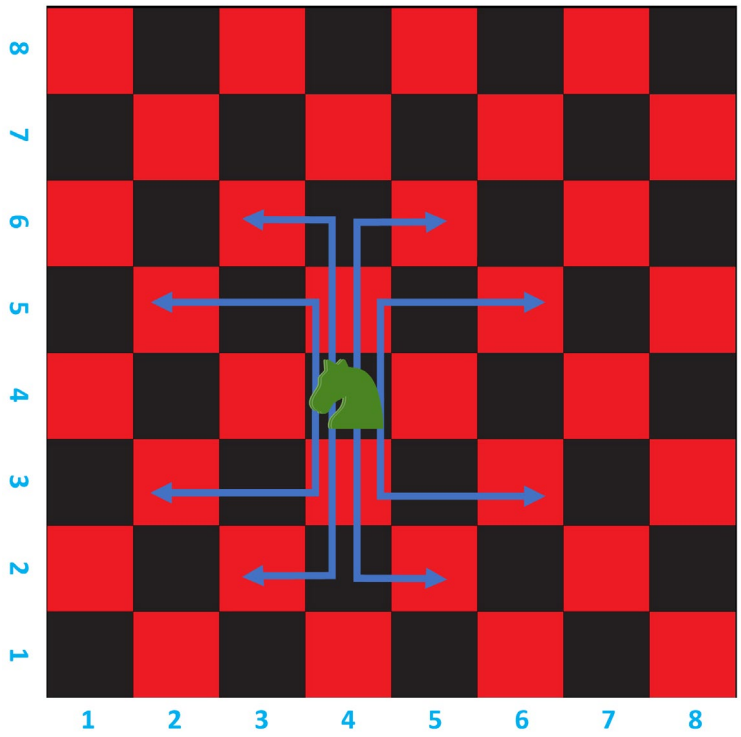
Input	Output
1 1 blue yellow 6 3 "bold text" "italics text" 2 1 bright dull 3 1 light dark 0	1. blue 2. italics text 3. dull 4. light

H. Super Knight

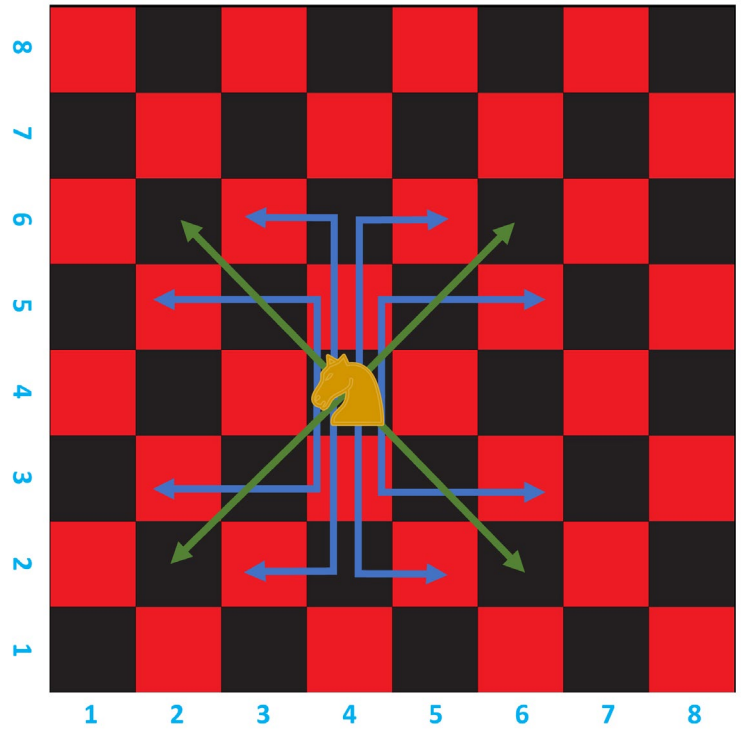
Program: knight. (cpp | java | py)

Description

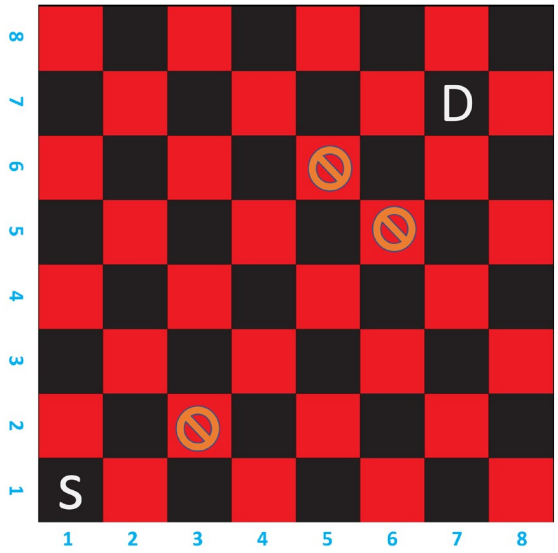
A knight has a maximum of 8 possible legal moves on a chess board. As seen in the figure below:



A super knight on the other hand has a maximum of 12 possible legal moves. It can also move two spaces diagonally, as shown in the figure below:



A board's configuration is defined by the source position (S), destination position (D), and forbidden squares (no-entry signs, where a piece cannot enter), as shown in the figure below:



A knight starting from the source will take 6 moves to reach the destination. Whereas as super knight will reach from the source to the destination in 3 moves. Your task is to write a program that given a 8x8 chess board with the provided source, destination, and forbidden positions finds the minimum number of moves required for a knight and super knight to go from the source to the destination. If no legal moves are possible then the program outputs -1.

Input

The first line of input starts with an integer **N** denoting the number of test cases.

For every test case, the first line contains two integers **x1**, **y1** representing the starting position. **1 >= x1, y1 <= 8**.

The second line contains two integers **x2, y2,** representing the destination. **1 >= x2, y2 <= 8.**

Next **K** lines contain a pair for forbidden positions Fx_i, Fy_i. **1 >= i <= K**. **1 >= Fx_i, Fy_i <= 8**.

The test case ends with a pair **-1-1.**

Output

For each input case output the minimum moves required for the knight and the super knight to move from the source to the destination on one line.

Sample Input / Output

Input
3 1 1 7 7 3 2 5 6 6 5 -1-1 8 8 7 7 7 6 6 7 -1-1 1 1 4 1 -1-1

Output
6 3 -1 3 3 2

I. Casting Budget

Program: cast. (cpp | java | py)

With the release of the 4th generation Marvel Movies taking place in the MCU, critics are saying that the mid-budget movie genre is now dead. As the predicted “next-next-next-next Martin Scorsese”, you aim to prove them wrong.

Taking note from the practices used in Theatre, you realize there is no good reason to not reuse a talented actor/actress in more than one role. For instance, if Mr. Programmer and Mr. Hero never appear together at the same time during the movie, one actor can play both roles!

It’s a good idea! After all, between make-up and CGI, nobody would be confused, and it’s a win-win: You get to slash the casting budget by at least half, and the actors/actresses get more chances of getting nominated during the awards season.

You tell your boss your idea, and he agrees, if you adhere to the following rules:

1. Male characters can only be played by male actors, and female characters can only be played by female actresses.
2. Each character must be played by one actor/actress throughout the whole movie. Changing the person who plays a given character during the movie is not allowed.
3. When two characters ever appear together at the same time during the movie, they must be played by different actors/actresses.

Given these restrictions, your job is to determine the **minimum** number of actors and actresses to determine the movie’s requirements.

Input

The input consists of several movie descriptions. Each description starts with one line that contains three integers **M, F, S** that specify the number of male (1<=**M**<=10) and female (1<=**F**<=10) characters occurring in the movie and the number of scenes that the movie has (1<=**S**<=100).
On the second and third line, the names of the male and female characters are given.
Then S lines follow, each line describing one scene. Each of these lines contains the number of people who occur in the scene and the list of their names.
The cases end with 0 0 0.

Output

Specification for each movie description, output three lines:

- a line saying: "Movie #n" where n is the number of the movie
- a line saying: "You need x actors and y actresses." where x and y is the solution. Use singular (actor, actress), if x = 1 or y = 1.
- a blank line after each case

Sample Input / Output

Input

```
1 1 1
Donald Daisy
2 Donald Daisy
4 3 6
Tarzan Jim John Tom
Lucy Cynthia Jane
3 Jim John Tom
2 Tarzan Lucy
2 Jane Cynthia
2 Jim Jane
2 Tarzan Jim
2 Tarzan Jane
0 0 0
```

Output

```
Movie #1
You need 1 actor and 1 actress.

Movie #2
You need 3 actors and 2 actresses.
```