Problem 1

Elo Rating Equation

The Social Network is a movie describing the story of the founders of the social-networking website, Facebook. For those of you who have seen the movie, you may remember Mark Zuckerberg inviting his friend Eduardo to give him his chess algorithm at the beginning of the movie when Mark creates Facemash.com. You may also remember he scribbles something like this on the window of his Harvard dorm room:

[(Total of opponents' ratings + 400 * (Wins - Losses)) / (Total Games)]

This is a variation of what is known as the Elo Rating Equation, and is a simple way to get an estimate of a PR (Performance Rating). In today's society, the Elo Rating system is an algorithm used by many ranking systems to predict the outcome of matches, and ensure a level of fairness between teams of different levels playing against each other. FIFA even uses the Elo rating system to rank soccer teams. Your job is to write a computer program that applies it to the top college football teams of a given season after at least seven games are played.

Input & Output

The first line of input contains an integer T, $(1 \leq T \leq 1000)$, which is the number of teams. The next T lines will contain team records, one line for each team. Each team record starts with the name of a team (a single string with no spaces), followed by a space, and the teams total wins and losses in the format `wins-losses` where wins and losses are both nonnegative integers with a single dash (-) character between them. All wins and losses are from teams in the same division (the PR ranking ignores games against the FCS division). This is followed by the rankings of each of the prior opponents with a single space between each ranking. You may assume that the number of rankings is always equal to the total games, and that no more than 15 games are played. Use the exact output format demonstrated below where each team is listed in descending order by PR. Round each PR to the nearest hundredth.

Sample Input

```
10
Alabama 7-1 5 77 29 2 92 100 51 33
Auburn 9-0 21 99 67 100 24 63 102 108 51
BoiseState 7-0 101 31 86 2 61 5 35
Nebraska 6-1 3 40 50 80 71 103 106
OhioState 8-1 22 85 53 7 82 43 103 47 4
Oklahoma 7-1 23 96 72 46 71 69 6 45
Oregon 7-0 0 33 52 109 13 49 89
TCU 8-0 86 98 54 21 31 39 72 11
Utah 8-0 69 11 0 5 69 31 21 72
Wisconsin 6-1 11 5 52 104 4 111 5
```

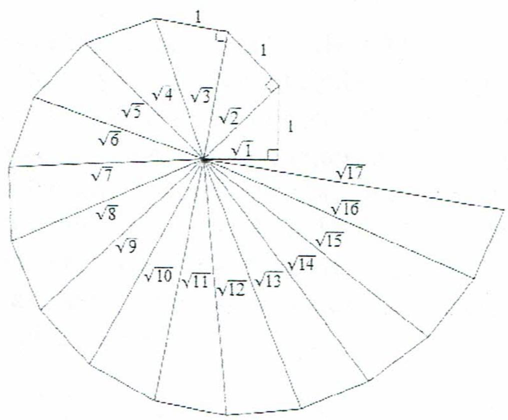
Output Corresponding to Sample Input

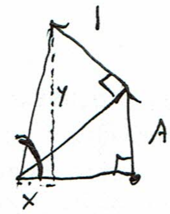
```
1)Auburn 470.56
2)TCU 451.50
3)Oregon 449.29
4)BoiseState 445.86
5)Utah 434.75
6)OhioState 360.67
7)Oklahoma 353.50
8)Nebraska 350.43
9)Alabama 348.63
10)Wisconsin 327.43
```

$\phi = \text{atan}\left(\frac{1}{\sqrt{n}}\right)$

Problem 2

Spiral of Theodorus





Theodorus of Cyrene was a Greek mathematician of the 5th century BC. A spiral of right triangles is named after him. This spiral is formed in the following manner.

The spiral is started with an isosceles right triangle, with each leg having a unit length of 1. The end of this first triangle's hypotenuse is (1,1). Another right triangle is formed, with one leg being the hypotenuse of the prior triangle and the other with length of 1. The process then repeats.

The spiral that Theodorus constructed stopped after the 16th triangle. You are to pick up where he left off. For a given positive integer n, you are interested in the segment whose endpoints are (0,0) and (a, b). This segment is one of the legs of the n^{th} triangle and the hypotenuse of the $(n\text{-}1)^{\text{st}}$ triangle. You need to output the ordered pair (a, b).

Input & Output

The first line of input contains an integer T, ($1 \leq T \leq 1000$), which is the number of test cases. Each test case will consist of a single positive integer n, $n \leq 10{,}000{,}000$. For each test case, there should be one line of output consisting of the x and y coordinates of the end of the n^{th} hypotenuse. Each coordinate should be rounded to four decimal places and have at least one digit to the left of the decimal place.

Sample Input

```
4
1
2
3
12
```

Output Corresponding to Sample Input

```
1.0000 0.0000    0°
1.0000 1.0000    45° 0.785398
0.2929 1.7071        1.40087335
0.3663 -3.4447       ~~-1.46495716~~

                     4.60644498145
```

Problem 3

Vigenère's Cipher

Vigenère's Cipher is a polyalphabetic cipher that uses the table below and a sequence of key numbers equal in length to the word being encrypted or decrypted. Unlike simple substitution ciphers like the Caesar's Cipher, Vigenère 's Cipher hides linguistic patterns and works by taking the key number as the row number and the plaintext letter to be encrypted as the column letter. The intersection of the row number and column letter provides the ciphertext. Close observation will reveal that the letter M is 5 letters after H, and the letter B is 23 letters after E if the alphabet is placed on a circular wheel, etc. Deciphering is done by taking the key number as the row number and finding the ciphertext letter on the row. Once the ciphertext letter is found, the column heading will reveal the plaintext letter.

Key	**Plaintext**	**Ciphertext**
5	H	M
23	E	B
13	L	Y
2	L	N
16	O	E

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
0	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
2	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
3	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
4	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
5	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
6	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
7	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
8	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
9	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
10	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
11	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
12	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
13	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
14	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
15	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
16	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
17	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
18	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
19	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
20	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
21	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
22	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
23	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
24	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	Y
25	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Input
The input is a sequence of test sets. Each set starts with a line containing a series of cipher key numbers where each number is separated by a space. These key numbers are followed by the letter `E` for encrypt or letter `D` for decrypt, followed by a space, and a message comprising only of uppercase and/or lowercase letters. No punctuation or spaces are ever used. The last set consists solely of the string "#" indicating end of input.

Each of the cipher key numbers will be a valid nonnegative integer n, $0 \leq n \leq 25$, for each corresponding alphabetic letter that must be either encrypted or decrypted. Assume that the number of key numbers matches the number of letters, and the length of each message is at least one character, but no more than 80 characters.

Output
For each message, output the encrypted or decrypted text message. The case of each letter in the output should match the case of the corresponding letter in the input.

Sample Input
```
1 E A
1 D A
5 23 13 2 16 E HELLO
5 23 13 2 16 D MBYNE
5 22 10 2 13 20 8 3 E Students
12 4 18 22 D Dsug
13 19 11 4 23 8 15 19 D PattYmae
#
```

Output Corresponding to Sample Input
```
B
Z
MBYNE
HELLO
Xpefrhbv
Rock
ChipBell
```

Problem 4

Australian Word Search

Andy is flipping through his Delta *Sky Magazine* on the long fifteen hour flight from Los Angeles to Sydney, and comes across a fun word search game. It is filled with lots of Australian lingo he would love to learn before arrival. It also makes him think a lot about how a computer might go about finding words in the grid.

Your job is to write a computer program that searches for words in the game. Words can be found horizontally or vertically in a 10 by 10 grid of characters, as well as along either of the two main diagonals that cross through the center of the grid. (The center of the grid has coordinates (4,4) where the first letter in the upper-left-hand corner has coordinates (0,0).)

Words can be found in any direction (left to right, right to left, up and down vertically, or moving either way along the two main diagonals). Each word is unique, and will be found only once in a given direction. All words are at least two characters long, and no more than ten.

Input

The input will start with ten lines each filled with exactly ten uppercase alphabetic characters, and no spaces. This is followed by the integer T, ($1 \leq T \leq 1000$), which is the number of words we are searching for within the grid. This is followed by T lines each containing a single word at the start. Each word will consist of all uppercase alphabetic characters, and no spaces.

Output

Your output will consist of T lines in the format below. Report back whether each word was found horizontally, vertically, diagonally, or not found at all in the format demonstrated below.

Sample Input

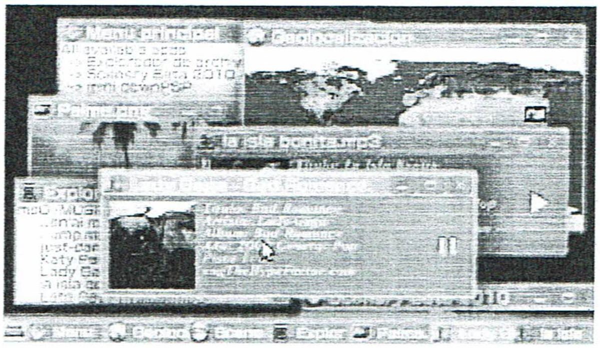
```
RGNAREMOOB
BEMQASAEAR
ZXESTSTIYO
OLMFEYEUAG
GLDINGOESI
IAKANOAROO
NRYAKKORDD
EOPZREEHLL
KOALAUHOSN
EIOUAEIOUP
7
BOOMERANG
OZ
REEF
MATE
HOON
DINGOES
```

Output Corresponding to Sample Input

```
BOOMERANG was found horizontally.
OZ was found vertically.
REEF was found diagonally.
MATE was found vertically.
HOON was found diagonally.
DINGOES was found horizontally.
JACKAROO was not found.
```

Problem 5

Windows



April is not very tidy with the desktop of her computer. She has the habit of opening windows on the screen, and then forgetting to close the application that created them. The result, of course, is a very cluttered desktop with some windows just peeking out from behind others and some completely hidden. Given that April doesn't log off for days, this is a formidable mess. Your job is to determine which window (if any) gets selected when April clicks on a certain position of the screen.

April's screen has a resolution of 10^6 by 10^6. When each window opens its position is given by the upper-left-hand corner, its width, and its height. (Assume position (0,0) is the location of the pixel in the upper-left-hand corner of her desktop. So, the lower-right-hand pixel has location (999999, 999999).)

Input

Input for each test case is a sequence of desktop descriptions. Each description consists of a line containing a positive integer n, the number of windows, followed by n lines, $n \leq 100$, describing windows in the order in which April opened them, followed by a line containing a positive integer m, the number of queries, followed by m lines, each describing a query. Each of the n window description lines contains four integers r, c, w, and h, where (r, c) is the row and column of the upper left pixel of the window, $0 \leq r, c \leq 999{,}999$, and w and h are the width and height of the window, in pixels, $1 \leq w, h$.

All windows will lie entirely on the desktop (i.e., no cropping). Each of the m query description lines contains two integers cr and cc, the row and column number of the location (which will be on the desktop). The last test case is followed by a line containing 0.

Output

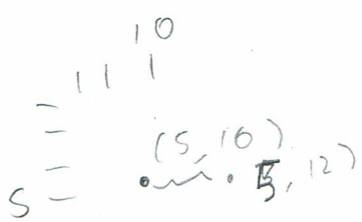
Using the format shown, print the desktop number, beginning with 1, followed by m lines, one per query. The i-th line should say either "window k", where k is the number of the window clicked on, or "`background`" if the query hit none of the windows. We assume that windows are numbered consecutively in the order in which April opened them, beginning with 1. Note that querying a window does not bring that window to the foreground on the screen.

Sample Input

```
3
1 2 3 3
2 3 2 2
3 4 2 2
4
3 5
1 2
4 2
3 3
2
5 10 2 10
100 100 100 100
2
5 13
100 101
0
```

Output Corresponding to Sample Input

```
Desktop 1:
window 3
window 1
background
window 2
Desktop 2:
background
window 2
```



Problem 6

The Chinese Zodiac

The Chinese Zodiac is a scheme that relates each year to an animal and its reputed attributes, according to a 12-year cycle. It is divided into 12 parts, and each part is associated with names of animals. The twelve animals are also linked to the traditional Chinese agricultural calendar, which runs alongside the better known lunar calendar. 2019 is the year of the pig.

Your job is to write a program that converts a given year to its corresponding animal in the Chinese Zodiac. Below is a table starting in the year 1965 along with its corresponding animal. 1965 was the year of the snake, and so were 1977, 1989, and 2001 given the 12-year cycle.

1965	snake
1966	horse
1967	sheep
1968	monkey
1969	rooster
1970	dog
1971	pig
1972	rat
1973	ox
1974	tiger
1975	rabbit
1976	dragon

Input
The first line of input contains an integer T, $(1 \leq T \leq 1000)$, which is the number of test cases. The next T lines will each contain a valid four digit year in the range from 1965 through 2165.

Output
There will be T lines of output, one per year. The general format of an output line is as follows.
<year input> `is the year of the` *<corresponding animal>*

Sample Input
```
4
1981
1970
2165
2011
```

Output Corresponding to Sample Input
```
1981 is the year of the rooster
1970 is the year of the dog
2165 is the year of the ox
2011 is the year of the rabbit
```