1 - Drop Zone

Evacuation missions and supply gathering missions are conducted on a regular basis by special teams. One of the first objectives of these missions is to set up a perimeter with barricades. Barricades are costly and take time to set up, so it is important to reduce the number of barricades necessary to block off an area.

You will be provided with several maps of high-interest drop zones. It is your job to write a program that will output the minimum number of barricades required for each drop zone.

Zombies will begin their approach from outside the map, thus any open area on the border is accessible to zombies. Barricades can be placed between any two adjacent open areas (including the drop zone) to block zombie movement. All zones outside of the map should be considered open areas.

Here is an example of one of the maps you will be provided:

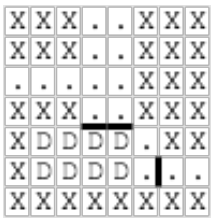
```
xxx..xxx
xxx..xxx
.....xxx
xxx..xxx
xDDDD.xx
xDDDD...
xxxxxxxx
```

Legend:

Symbol	Description
x	Impassible, zombies cannot move through these areas
.	Open area, zombies can move through these zones, but not diagonally. Barricades may be placed between these areas.

(period)	
`D`	Drop zone, this zone must be protected at all costs. Barricades must effectively block zombie movement from the edges of the map to this zone. This area is treated like an open area for zombie movement and barricade placement. There will be exactly one contiguous drop zone in every map. Drop zones may be on the edges of the map.

Example barricade configuration:



Solution: 3

Input

The first number, `N` (`1 <= N <= 20`), will be the number of maps. For each map, there will be two parameters, `R` and `C` (`1 <= R, C <= 150`), denoting the number of rows and columns in the map, followed by the map itself as described above. Each row will be on its own line and all rows will have the same number of characters equal to `C`.

Output

For each map, print a single line containing the minimum number of barricades required to block off the drop zone.

Sample Input

```
2
7 8
XXX..XXX
XXX..XXX
.....XXX
XXX..XXX
XDDDD.XX
XDDDD...
XXXXXXXX
```

```
5 5
XX.XX
.DDD.
XD.DX
X....
X....
```

Sample Output

```
3
6
```

2 - Effective Infection Time

You are estimating the threat level of quarantined zones that have been abandoned to the infection. One of the key variables in determining a zone's threat level is the EIT (Effective Infection Time). This information is essential for planning strike dates to reclaim quarantined zones. The EIT is calculated according to the following rules:

- The EIT is the result of a function of two dates: The infection date and the strike date.
- All years are in A.Z. (After Zombie).
- Every month counts for a fraction of an EIT after its last day has passed. This means the month of the strike date does not count for EIT.
- The first calendar year of the infection is calculated as 1/2 EIT.
    - If the end of the year is not reached, each month only counts for a fraction of the 1/2 EIT. If a zone was infected in January of the first year, then the 1/2 EIT is spread across 12 months ((1/2)/12 = ~ 0.0417 EIT per month). If a zone was infected in March of the first year, then the 1/2 EIT is spread across 10 months ((1/2)/10 = 0.0500 EIT per month).
    - If the end of the year is reached, the year counts as a full 1/2 EIT, regardless of the infection month. In other words, a zone infected in February of 15 A.Z. counts as only 1/2 (one-half) EIT *after* December 15 A.Z. A zone infected in December of the same year will also count as 1/2 EIT.
- All following years are calculated as 1 EIT. Each calendar month, beginning with January, counts for 1/12 EIT (~ 0.0833 EIT).
- Every zone infected on the same month will have the same EIT for any given strike date. Therefore only the month and year are given.

The number and order of months in a calendar year remains the same as the modern Gregorian calendar.

Input

The first line will be an integer ***N***, where 1 <= N <= 50 giving the number of zones. For each zone, a pair of lines of will be provided:

- The first line contains the infection date. The second contains the strike date.
- The first integer of a date represents the month, `M` (`1 <= M <= 12`), and the second integer represents the year, `Y` (`0000 <= Y <= 0030`). The year will always have 4 digits.
- The strike date will never precede the infection date.

Output

Output the EIT for each zone on its own line. The EIT must be rounded to the fourth digit after the decimal point. The ones-digit must always be printed even if it is a zero.

Sample Input

```
2
2 0009
11 0012
3 0010
10 0010
```

Sample Output

```
3.3333
0.3500
```

3 - Outpost Navigation

The city has been overtaken by zombies, and you and your team are running low on supplies. Through careful reconnaissance and planning, you have been given a map with a list of safe outposts along with a report detailing the number of zombies that have been spotted in the area. Each outpost is part of a radio network and uses its radio call sign as the name for its location.

Your team has a limited supply of ammunition and can only hold off zombie groups of a certain size. Each unit of ammunition can hold off a single zombie per trip. In order to travel a road safely, you must spend an amount of ammunition equal to the number of zombie encounters on the road. Given the reconnaissance data, determine a safe path to an outpost that has the supplies that you need with the least number of zombie encounters. You will begin your journey from the first listed outpost and will start with any ammunition at that outpost. There may be *up to one other* outpost that has ammunition but not the supplies you need. If you travel to that outpost, you may collect all of its ammunition. *Any number* of outposts may have supplies that you need, but you only need to travel to one of them.

If for any reason you need to travel a road multiple times, you will encounter the same number of zombies as you did on your first trip for each subsequent trip, and the zombies do count as additional zombie encounters. Zombies are are not easily killed with bullets.

Input

The first line of input will contain the number of test cases, `C` (`1 <= C <= 50`).

For each test case, there will be a single line containing the number of outposts, `N` (`1 <= N <= 100`), followed by the number of roads, `R` (`0 <= R <= 200`).

```
1 1
```

For each outpost, there will be a single line containing the call sign (6 characters from `A-Z` and `0-9`), followed by the amount of ammunition at the outpost `A`, (`0 <= A <= 100`), followed by an indication of whether the outpost has the supplies you need ("`yes`" or "`no`").

```
CLLSGN 0 yes
```

For each road, there will be a single line containing the call signs of the 2 outposts it connects followed by the number of zombies you will encounter on the road, `Z` (`0 <= Z <= 100`).

```
OTPST1 OTPST2 5
```

Output

For each test case, print a single line containing the minimum number of zombie encounters for a safe path to an outpost with the supplies you need. If there is no safe path to supplies, print a single line containing the string "`No safe path`".

Sample Input

```
3
3 3
OR1G1N 5 no
NUL000 0 no
T4RG3T 0 yes
OR1G1N T4RG3T 20
OR1G1N NUL000 1
NUL000 T4RG3T 1
3 2
OR1G1N 1 no
NUL000 0 no
T4RG3T 0 yes
OR1G1N NUL000 1
NUL000 T4RG3T 1
3 2
OR1G1N 1 no
AMMUN1 1 no
T4RG3T 0 yes
OR1G1N AMMUN1 1
AMMUN1 T4RG3T 1
```

Sample Output

```
2
No safe path
2
```

4 - Multikill

The Zombie Apocalypse is here! Zombies are quite easy to kill; however, ammunition is scarce so we need to maximize the explosive potential. Our most powerful defensive weapon is an automated grenade launcher that can identify zombified targets at extreme distances. The one missing piece is the code for optimizing the blasts to hit multiple targets.

For each known group of zombies, you will be given a kill radius and the locations of the zombies. The program should identify a coordinate to hit with the explosive to kill as many zombies as possible and output the maximum number of targets that can be killed with one round. A zombie will be killed if its distance from the explosive is equal to or less than the kill radius.

Input

The first line of input will contain an integer representing the number of test cases, `C` (`1 <= C <= 20`). For each test case, there will be a single line containing a kill radius as a real number, `R` (`0 < R <= 1000.0`), and a count of zombie targets, `N` (`0 <= N <= 25`), followed by `N` lines containing a pair of Cartesian coordinates `X Y` (`-10^6 <= X, Y <= 10^6`) giving each zombie position. The units used for the kill radius and the coordinates are both in meters.

```
C
R N
X0 Y0
...
XN-1 YN-1
```

Output

Output will be a single line per test case with the number of targets that can be killed with a single explosive round. Output for each test case should be on its own line.

```
K0
...
```

Sample Input

```
2
3 5
1 0
0 0
0 1
1 1
5 5
1.0 1
1.0 0.0
```

Sample Output

```
4
1
```

5 - Virology

Little is understood about the virus and the way it infects its human hosts, but what has been discovered is a peculiar pattern in human DNA that can tell virologists if a particular person is vulnerable or immune to the virus. DNA was sampled from every individual working with the CDC, and a pattern was recognized to be present in only those who were infected by the virus.

We have isolated 9 genes within human DNA that can tell us if a person is vulnerable to infection. Most people have at least 14 occurrences of these genes. An individual that is considered **vulnerable** is known to have met the following conditions:

- The individual must have exactly 14 occurrences of the numbered DNA genes (1-9). You will only test samples from people that meet this condition. There will be a max of 3 of the same gene in a test case. The order of the genes is not significant.
- Within their DNA there must be 4 total instances of triples and/or straights and 1 pair
    - triples (3 of the same gene, example: `[7 7 7]`
    - straights of 3 (examples: `[1 2 3]` `[7 8 9]` `[4 5 6]`)
    - pair (2 of the same gene, example: `[9 9]`)

    Note: An instance of a gene cannot be reused in multiple sets

So if an individual with 14 numbered genes in their DNA has

- 4 triples/straights AND
- 1 pair

then they are vulnerable. If this pattern cannot be found in an individual's DNA then that individual is **immune**.

Your job is to take a list of DNA samples from individuals with 14 of the numbered genes and determine if they are vulnerable.

Input

The first number will be the number of test cases `N` (`1 <= N <= 200000`). Following that will be `N` lines of 14 numbers separated by spaces indicating the genes present in the DNA.

Output

If an individual is vulnerable to the virus, output `Vulnerable`. Otherwise, output `Immune`. Output each answer on a separate line.

Sample Input

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

Sample Output

```
Vulnerable
Immune
```

6 - Worst Case Scenario

The disease is spreading quickly, and the government has requested a map of infected regions. They would like to see what certain regions would look like, worst-case scenario, given some series of events. There are four stages of infection that the feds are interested in.

Stage A

The infection has not yet reached this zone. It is still possible for such zones to advance in infection stage due to an infection event.

Stage B

This zone is in the early stages of infection. There are reports of erratic behavior in individual citizens, quarantines are starting, and the municipal authorities still have things under control.

Stage C

This zone is dealing with mobs of zombies in a few areas. Quarantines are occurring on the scale of towns and small cities. The infection is contained to the zone but growing out of control.

Stage D

This zone has lost control of its borders. It is only a matter of time before the infection spreads to neighboring zones.

An infection event is anything that causes a zone to advance towards a higher stage of infection. This is usually the result of a quarantine breach, virus mutation, or biological terrorism. When an infection event occurs in a zone, it will, worst-cast scenario, cause the zone to advance in stage. A zone in stage D cannot advance in stage any further and will, worst-case scenario, result in an outbreak for each additional infection event it experiences. When an outbreak occurs, all neighboring zones experience an additional infection event. An outbreak can cause a chain reaction of outbreaks, but an outbreak can only cause a maximum of one outbreak in each zone.

Zones marked with an `x` character are impassible and unpopulated areas that cannot become infected. An adjacent zone is one zone above, below, to the left, or to the right on the grid of zones. During an outbreak, the infection cannot spread diagonally.

You will receive a grid of zones with their starting stages and a sequence of infection events with the coordinates of the zones in which they occur. Each infection event should be processed before

calculating the result of the next infection event.

Input

The first line of input will contain the number of test cases, `N` (`1 <= N <= 50`). Each test case will begin with a line giving the width, `W` (`1 <= W <= 100`) and height, `H` (`1 <= H <= 100`) of the grid, followed by the grid itself. Each zone in the grid will be marked by the stage that the zone is currently in, or the character `X` if the zone is impassable.

Following the grid will be a line with the number of infection events `I` (`0 <= I <= 1000`). `I` lines will follow each giving an `X` and `Y` coordinate of a zone. Each line represents an infection event that occurs on the `Xth` column (the left most column being `X` = 0) and the `Yth` row (the top row being `Y` = 0).

Output

Your output should show each grid with its states if the worst-case scenario occurs.

Sample Input

```
2
3 4
AAA
AXA
XAA
AAA
5
1 2
2 2
1 2
1 2
1 2
7 6
AAAAAAA
AAAAADA
AAXAAAA
AAAXDAA
AAXXAAB
AAXXACD
10
```

```
6 4
4 4
4 4
6 3
4 3
6 4
6 2
6 4
6 4
6 4
```

Sample Output

```
AAA
AXA
XDC
ABA
AAAAAAA
AAAAADA
AAXACBC
AAAXDDD
AAXXDDD
AAXXDDD
```

7 - Zombie Invasion

A group of survivors has arrived by helicopter to an isolated island. The island is made up of a long narrow strip of villages. The infected survivors arrived in the village to the far east and accidentally infected the native population. The islanders are now attempting to escape the zombies that have appeared on the east coast.

You are given `N` cases with 20 non-negative integers that represent the number of islanders at a given village. The villages are represented from west to east (left to right, respectively), with the zombies moving in from the east. The islanders have peculiar customs for traveling and will only move between villages in pairs. Curiously, for every pair that travels between two villages, only one of them ever survives the trip. As the zombies move west, islanders will travel to the village immediately west of their current village as long as there are at least two islanders there. If there are an odd number people in a village then one stays in the village and the rest move to the next village in pairs. Once the islanders reach the village on the west coast, they will stop traveling.

Determine how many islanders remain at each village and the number that make it safely to the village on the west coast (far left).

Input

The first line of data represents the number of data sets you will read in, `N` (`1 <= N <= 50`).

There will then be `N` lines of twenty 20 non-negative integers each. Each integer (`<= 1000`) represents the number of islanders who reside in a village. The leftmost integer represents the village on the west coast, and the rightmost integer represents the village on the east coast.

Output

Your output will be `N` lines of twenty 20 non-negative integers. The left most number will represent the number of islanders that reached the west. Each number to the right will represent the number of people that stayed behind in each village.

Sample Input

```
1
0 0 0 0 77 0 0 99 0 0 0 40 0 0 0 17 0 1 13 10
```

Sample Output

```
5 1 0 0 1 1 0 1 1 0 0 1 0 0 1 1 1 0 0 0
```

8 - Zombdar

You are reading data from a zombie sensor. The sensor scans the area to obtain the number of zombies in the immediate area. The zombie sensor normally writes log entries in the form of "`Zombies: <integer>;`" or "`No Zombies;`" to its buffer as it performs scans, but it may also write "`RUN;`" when the sensor is overloaded. These are the only values that will be written to the buffer.

The zombie sensor's serial port emits a line containing whatever data is in its buffer every second, regardless of whether the buffer contains a complete log entry, or even multiple entries.

A valid sequence of log entries may be:

```
Zombies: 5;
Zombies: 1;
No Zombies;
Zombies: 70;
RUN;
RUN;
RUN;
```

But the sensor's serial port may emit:

```
Zom
bies:
 5;Zombies: 1
;
No Zombies;
Zombies 70;
RUN;
RUN;RUN;Zo
```

It is imperative to process the serial port data correctly if you are to survive.

Input

The first line of input contains the number of data sets, `N` (`1 <= N <= 50`). For each data set, the input contains the raw data emitted by the zombie sensor's serial port (see above for details) followed by a line containing only the string "`END OF CASE`". Since data is emitted by the zombie sensor's serial port once per second, the first line of input is read after 1 second, the 2nd line after 2 seconds, and so on.

Output

For each complete log entry, you should output a line containing "`timestamp: log_entry`", where `timestamp` is the number of seconds elapsed between the start of the data set and the time at which the entry was completely parsed.

Sample Input

```
2
Zom
bies:
 5;Zombies: 1
;
No Zombies;
Zombies: 70;
RUN;
RUN;RUN;RU
END OF CASE
No
 Zombies;
No
 Zombies;
Zombies: 4;Z
ombies
: 14;
Zombies
: 60;
Zombies:
 100;

Zom
bies: 15;
RUN;
```

```
RUN;RUN;
R
END OF CASE
```

Sample Output

```
3: Zombies: 5;
4: Zombies: 1;
5: No Zombies;
6: Zombies: 70;
7: RUN;
8: RUN;
8: RUN;
2: No Zombies;
4: No Zombies;
5: Zombies: 4;
7: Zombies: 14;
9: Zombies: 60;
11: Zombies: 100;
14: Zombies: 15;
15: RUN;
16: RUN;
16: RUN;
```