AddThemUp
Input File: AddThemUp.txt

A sequence of numbers is defined recursively as:

The first number in the sequence, $N_1 = 1$
The second number in the sequence, $N_2 = 2$
Any other number in the sequence ($i \neq 1$ or 2), $N_i = N_{i-2} - N_{i-1}$

Thus, the first eight numbers in the sequence are 1, 2, -1, 3, -4, 7, -11, 18.

Write a program to calculate the ***sum*** of the first n terms in this sequence, for $1 \leq n \leq 46$.

Inputs
The first line will contain the number of sums to calculate. This will be followed by the number of terms in the sequence that will be included in the sum.

Outputs
Output the sums of the terms of the sequence, one sum per line.

Sample input
```
5
3
6
9
12
46
```

Sample output
```
2
8
-14
80
969323033
```

ChooChoo
Input File: ChooChoo.txt

Your math professor, Barbara, has retired from teaching to become a professional golfer. Until she establishes herself on the LPGA tour, she travels by train to the tournament sites each week. Unfortunately, she has missed several tee times because her six-digit train number was misread by the traffic controllers, and her train was routed to the wrong destination. To remedy this situation, she has proposed that a seventh digit be added to the train number so that the traffic control computer can verify that the traffic controller has properly entered the train number into the computer.

To verify the train number the computer adds the first, third, fifth and seventh digits. Then it adds the remaining digits, and doubles their sum. Finally, the two sums are added together. If the result is an even multiple of 7, there is a high probability the train number was typed properly and Professor Barbara will make her tee time.

Write a program to verify that a train number was properly entered into the traffic control computer.

Inputs
The first line will contain the number of trains whose numbers are to be verified, n. This will be followed by n lines of input, one seven digit train number per line.

Outputs
For each train number to be verified there will be one line of output containing two items separated by one space. The first item on the line will be the seven digit train number. This will be followed by the words “valid” or “invalid” depending on the result of the verification algorithm.

Sample inputs
4
153-524-8
134-824-3
143-924-0
127-323-2

Sample output
153-524-8 valid
134-824-3 invalid
143-924-0 invalid
127-323-2 valid

Gophers
Input File: Gophers.txt

Two gophers, Logan and Evie are grazing in a field that contains several gopher holes. Since gophers do not like to get wet, when it begins to rain they head for the nearest hole. The locations of each hole has been mapped using a polar coordinate system (r, Θ), and Logan and Evie always carry their cell phones with them that transmits their location to a GPS system in the same polar coordinate system. The azimuth angle, Θ, of the system is measured in degrees and is positive in an anticlockwise direction, and r is measured in meters. Normally, Evie and Logan run at the same speed, 3 meters per second. However, their speed is cut in half for every gram of food they eat. Your task is to determine which gopher will reach shelter first when it begins to rain.

Inputs

The first line of input will contain the number data sets. Each data set will contain three additional lines. The first line of a data set contains the number of gopher holes followed by the polar coordinates (r followed by Θ) of each hole. The second line of a data set contains Logan's location (r followed by Θ), followed by the grams of grass Logan has eaten. The third line of a data set contains Evie's location (r followed by Θ), follow by the grams of grass Evie has eaten. All inputs are integers except for the grams of grass the gophers consume.

Outputs

Output one line per data set that contains name of the gopher that was the first to enter a hole. If Evie and Logan arrive at the hole at the same time, output "Ladies first".

Sample inputs

```
3
3 54 180 135 45 70 290
10 35 2.0
300 90 0.0
1 100 0
100 90 0.0
100 90 0.0
10 200 45 190 270 54 180 135 45 70 290 10 300 200 55 500 75 90 285 100 0
91 285 7.0
60 140 1.8
```

Sample output

```
Evie
Ladies first
Logan
```

Lucky
Input File: Lucky.txt

The lucky numbers between 1 and n are a set of integers between 1 and n (inclusive) that are "lucky" enough to survive the following screening process:

> 1 is a lucky number. The next integer encountered between 1 and n is 2, so every second number between 1 and n is eliminated (2, 4, etc.). The next integer encountered in the resulting set of numbers is 3, so every third number is eliminated (5, 11, etc). The next integer encountered in resulting set of numbers from 1 to n is 7, so every seventh number is eliminated (19, 31, etc). The screening process continues in this way until no more numbers can be eliminated.

If n were 23, the following would be the screening process to generate the lucky numbers;

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
1   3   5   7   9    11    13    15    17    19    21    23
1   3       7   9          13    15          19    21
1   3       7   9          13    15                21
1   3       7   9          13    15                21
```

Inputs
The first line will contain the number of lucky number sets to generate, g. The next g lines will give the maximum possible lucky number, n, in each of the data sets.

Outputs
There will be two lines of output for each set of lucky numbers. The first line will contain the maximum possible lucky number for the data set. The next line will contain the lucky numbers in that data set in ascending order, with each number separated by one space.

Sample inputs
```
2
23
60
```

Sample output
```
23
1 3 7 9 13 15 21
60
1 3 7 9 13 15 21 25 31 33 37 43 49 51
```

NCAA
Input File: NCAA.txt

The NCAA Division 1 baseball playoffs is about to begin. It is a single elimination tournament, one loss and you go home. St. Joseph's College (SJC) had a good team last year, but we lost to the University of Southern California (USC) in the finals.

The NCAA has adopted a String notation used to keep track of the winners and losers, and which teams play each other. As part of the notation each team is assigned a unique abbreviation (e.g., `SJC` is the abbreviation for St. Joseph's College, `USC` for University of Southern California). Using theses abbreviation and the letters `L` for lost and `W` for win, the NCAA notation `SJC USC W` or `USC SJC L` would both indicate that the University of Southern California defeated St. Joseph's College. Similarly, the notation `SJC USC PEN L W` and `SJC USC W PEN L` would both indicate that `USC` was the winner of a three team tournament between `SJC`, `USC` and `PEN`.

Write a program to determine the winner of the NCAA baseball championship given the NCAA String notation.

Inputs
The first line will contain the number of tournaments to process. This will be followed by one line per tournament that contains the NCAA String notation for the tournament.

Outputs
For each tournament output the winner, one tournament winner per output line.

Sample inputs
```
4
SJC USC PEN L W
SCJ USC W PEN L
SJC USC PEN L W PSU W FSU L
NCS USC PEN SJC W USC PEN W L W W
```

Sample output
```
USC
USC
PSU
SJC
```

TimeDilation
Input File: TimeDilation.txt

Your older sister Nadia is part of the NASA deep space exploration team. As such she will be making fly-bys to stars and planets (e.g., Proxima Centauri, the closest star to our sun). The space ship she will use is a hyper-vehicle able to travel at speeds close to the speed of light. You would like to throw her a birthday party when she returns from her trip, but you are uncertain how old she will be when she returns because of a counterintuitive phenomenon called "time dilations".

Time dilations occur when traveling near the speed of light. The result of these dilations is to reduce the aging process of astronauts during their journey. Specifically, given the number of earth years a journey will take, N_J, we can calculate the number of years older an astronaut will be at the end of the journey, N_A, using the below formula:

$$N_A = N_J * \sqrt{(1 - v^2/c^2)}$$

where: v is the speed of the space ship, and c is the speed of light. So, if Nadia traveled on a 20 year journey at 0.995 times the speed of light the above equation reveals that she would only have only aged 2.0 years during the trip. If you were 19 and she was 25 when she began the journey, you would be 39 and she would be 27 when she returned. She would have become your *younger* sister!

Write a program to determine the number of years an astronaut will age on a given mission.

Inputs
The first line of input will contain the number of missions to process followed by the speed of light, your current age and Nadia's current age. This will be followed by one input line per mission that contains the ratio of the speed of the space ship to the speed of light, followed by the time (in earth years) for the round trip mission.

Outputs
For each mission output your age and Nadia's age when Nadia returns from the mission, followed by the speed of light and the speed of the space ship. Each mission's output should be on a separate line *formatted and annotated* (with commas and numeric precision) ***exactly*** as shown below.

Sample inputs
```
3 186282.27860 19.0 25.0
0.99500 20.0
0.10000 20.0
0.99990 20.0
```

Sample output
```
I will be 39.0 and Nadia will be 27.0, assuming: c = 186,282.28 v = 185,350.87
I will be 39.0 and Nadia will be 44.9, assuming: c = 186,282.28 v = 18,628.23
I will be 39.0 and Nadia will be 25.3, assuming: c = 186,282.28 v = 186,263.65
```

Tolls
Input File: Tolls.txt

Each square of a 10x10 checkerboard has a toll associated with it that must be paid when you enter the square. You wish to travel from the bottom most row to the top most row and

minimize the total of the tolls along the way. Write a program to output the row and column numbers of a route that minimizes the tolls. When making a move, you must stay on the board, the row number *must* increase by 1 and the column number can change by -1, 0, or +1. Tolls range from 0 to 9, and row 1 and column 1 is the lower left most square of the checkerboard.

	column 1	2	3	4	5	6	7	8	9	10
10	6	4	7	4	8	3	6	7	2	4
9	9	1	4	7	3	6	8	6	1	4
8	4	8	1	9	7	9	2	3	5	4
7	1	8	6	6	8	4	8	3	8	2
6	7	3	7	4	4	1	5	9	9	4
5	1	6	3	2	1	4	3	3	7	9
4	5	3	8	4	2	6	7	9	3	5
3	6	4	3	8	7	1	2	4	7	4
2	8	8	3	6	5	8	3	9	1	5
row 1	0	3	5	6	1	2	7	1	9	4

Inputs:

The tolls associated with each square of the checkerboard, one line per row. The first line of input is the tolls associated with row 10, the second line the tolls associated with row 9, etc. The tolls on a line will be separated by a space.

Outputs:

The minimum total toll followed by 10 lines that give the row and column numbers of the minimum toll path through the checkerboard. Each square's location will be on a separate line, with the row number preceding the column number. There will be a space between each row and column number.

Sample Input:

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

Sample Output:

```
23
1 8
2 7
3 6
```

```
4 5
5 5
6 6
7 7
8 8
9 9
10 9
```