1- Breakfast Club
Input File: BreakfastClubIn.txt

Gretchen and her friends have formed a breakfast club, and every Sunday morning she and her friends eat breakfast at her house. Since she has a chicken farm, each person's breakfast includes three eggs. In preparation for the feast, Saturday afternoon she gathers enough eggs for the Sunday breakfast, and an additional two eggs for each of her breakfasts during the remainder of the week. In addition, she also gathers five times that total amount to sell at her farm stand.

For example, if she has four friends (and herself) at her Sunday breakfast she would have to gather 162 eggs on Saturday: 15 for Sunday's five person breakfast, 12 for the remainder of the week, and 135 for the farm stand. Your task is to determine how many eggs she has to gather on Saturday afternoon, given the number of friends that will join her for breakfast the next day.

Inputs:
There will be one line of input that contains the number of friends that will join Gretchen for breakfast, an integer.

Outputs:
There will be one line of output that contains the number of eggs she will collect on Saturday afternoon in preparation for Sunday's breakfast, annotated exactly as shown below.

Sample Inputs
```
4
```

Resultant Output
```
Gretchen will gather 162 eggs on Saturday
```

2- Potatoes
Input File: PotatoesIn.txt

Potatoes are sold in 2, 5, 10, 20 40, and 60 pound bags. Vikki was on her way to the checkout counter when she realized she needed potatoes. Since she only had one hand free, she could only pick up one bag of potatoes and then proceed to the checkout counter. Given the pounds of potatoes she needs, determine which size bag she should pick up so she can be sure she has all the potatoes she needs, even if she has to purchase a few extra pounds. In the event she needs more potatoes than the 60 pound bag contains, she would purchase the 60 pound bag.

Inputs:
The first line contains a positive integer indicating how many shopping trips to consider. This is followed by one line per shopping trip containing one integer that represents the pounds of potatoes Vikki needs.

Outputs:
For each shopping trip there will be one line of output that contains two integers separated by a space. The first integer is the weight of the bag Vikki should purchase. The second integer is the extras pounds of potatoes she purchased: positive if the bag contains more potatoes that she needs, zero if it contains the exact amount of potatoes she needs, and negative if she needs more than 60 pounds of potatoes.

Sample Inputs
```
5
15
33
64
60
1
```

Resultant Output
```
20 5
40 7
60 -4
60 0
2 1
```

3- Caesar
Input File: CeasarIn.txt

Having heard about a quantum computer's ability to break the scheme used to encrypt messages sent over the internet, Julius has formulated a scheme to pre-encrypt messages he emails to Rome.

He will randomly choose an integer encryption key, **`K`**, whose range is $-50 \leq$ **`K`** ≤ 50, and use it in his encryption scheme. Each letter in the message will be shifted circularly **`K`** letters in the alphabet. For example, if **`K`** were chosen to be 3, all occurrences of the letter 'A' in the email would be changed to the letter 'D', and occurrences of the letter 'Y' would be changed to the letter 'B'. If **`K`** were -3, the letter 'A' would be changed 'X', and 'Y' would be changed to 'V'.

In addition, after circularly shifting the letters in the message, all upper case letters in the message would be changed to lower case, and lower case letters would be changed to upper case. All non-alphabetic characters in the message are not changed. Then the email would be sent.

Thus, the message: `What? Wow, he fiddled while everything burned!`
would be changed to: `tEXQ? tLT, EB CFAAIBA TEFIB BSBOVQEFKD YROKBA!`
For an encryption key, **`K`**, equal to -3.

Inputs:
The first line contains a positive integer indicating how many messages are to be encrypted. This is followed by two lines per message. The first line contains one integer that represents the value of the encryption key, **`K`**. The second line contains the message to be encrypted.

Outputs:
For each input message, the program should generate a single line of output that contains the encrypted message.

Sample Inputs
```
2
-3
What? Wow, he fiddled while everything burned!
32
CdEfG, & hIjKlMnOp, + QrStUv? WxYzAb.;
```

Resultant Output
```
tEXQ? tLT, EB CFAAIBA TEFIB BSBOVQEFKD YROKBA!
iJkLm, & NoPqRsTuV, + wXyZaB? cDeFgH.;
```

4- Rock Loader

Input File: RockLoaderIn.txt

Rocks, that weigh 1, or 2, or 3, or ..., or 30,000 pounds, are stored on a quarry warehouse shelf in increasing weight order from left to right. When contractors order a shipment of rocks, they specify the total weight **`W`** of rocks the want in the shipment, which always contains at least two rocks. In addition, a shipment always contains rocks whose weights are sequential (e.g., 2, 3, 4), and whose combined weight is **`W`** (e.g., 9).

Ray uses a bucket loader to scoop the sequentially weighted rocks, whose total weight is **`W`**, off the shelf and then place them in the contractor's truck. Your task is to determine the number of possible sequential rock weight groupings, **`n`**, Ray can choose from and the total of the weights of the lightest rock in each grouping, **`T`**.

For example, if the contractor orders 15 pounds of rocks there are three (**`n`** = 3) possible sequential rock weight groupings:

Grouping 1	Grouping 2	Grouping 3
1, 2, 3, 4, 5	4, 5, 6	7, 8

and the total weight of the lightest rock in each of the three groupings, **`T`**, is 12 = 1 + 4 + 7.

Inputs:

The first line contains a positive integer indicating how many warehouses to consider. This is followed by line of input per warehouse, that contains one integer representing the total weight **`W`** of the rock shipment.

Outputs:

For each warehouse there will one line of output that contains two integers separated by a space. The first of these integers will be the number of sequential rock weight groupings, **`n`**, each of whose rock weight sum adds up to the total weight **`W`** of the rock shipment. The second integer will represent the sum of the weights of the first rock in each grouping.

Sample Inputs

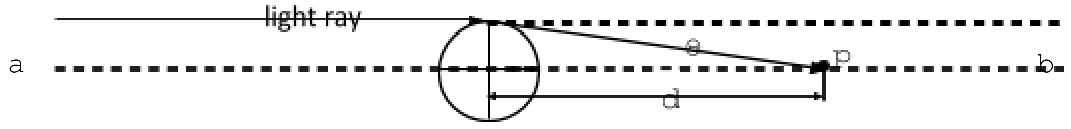
```
4

3

5

15

26754
```

Resulting Outputs

```
1 1

1 2

3 12

15 28351
```

5- Relatively Bent
Input File: RelativelyBentIn.txt

In 1919 Albert Einstein's formulated his theory of general relativity, which predicts that a ray of light grazing the surface of a massive spherical object would be bent through an angle Θ as shown below. This would permit a person whose eye is on the line ab, which is parallel to the initial path of the light ray and through the center of the spherical object, to see the light if the person's eye is positioned at point p.



Furthermore, Dr. Einstein's theory predicted that the angle Θ could be calculated in radians as:

$$\Theta = 4*G*M/(r*c^2) \text{ radians}$$

where G is the universal gravity constant , M is the mass of the spherical object and r is its radius, and c is the speed of light. In metric (mkgs) units: $G = 6.7x10^{-11}$ and $c = 3x10^8$, and assuming the spherical object is our sun, $M = 2x10^{30}$ and $r = 7x10^8$.

The radius of the sun in Mr. Spock's solar system, in which they also use the metric system, is nr times our sun's radius, and its mass is nm times our sun's mass. Given nr and nm, you have two tasks:

1. determine the density of Mr. Spock's sun, calculated as $M / (\frac{4}{3}\pi r^3)$, in mkgs units, and
2. determine the distance d, the number of Spock's sun's radii from the sphere's center along line ab to the point p, where he should position his eye in order to see the bent a light.

Inputs:
The first line contains a positive integer indicating how many solar systems there will be. This is followed by one line of inputs per solar system, that contains two real numbers separated by a space. The first of these numbers is the ratio of the solar system's sun's mass to the mass of our sun, and the second integer is the ratio of the solar system's sun's radius to our sun's radius.

Outputs:
For each solar system there will one line of output that contains two real numbers separated by a space. The first number will represent the density of the solar system's sun, and the second number will be the number of Spock's sun's radii from the center of the solar system's sun to the observer's eye location (the point p). Each output number will contain two digits of precision rounded up.

Sample Inputs

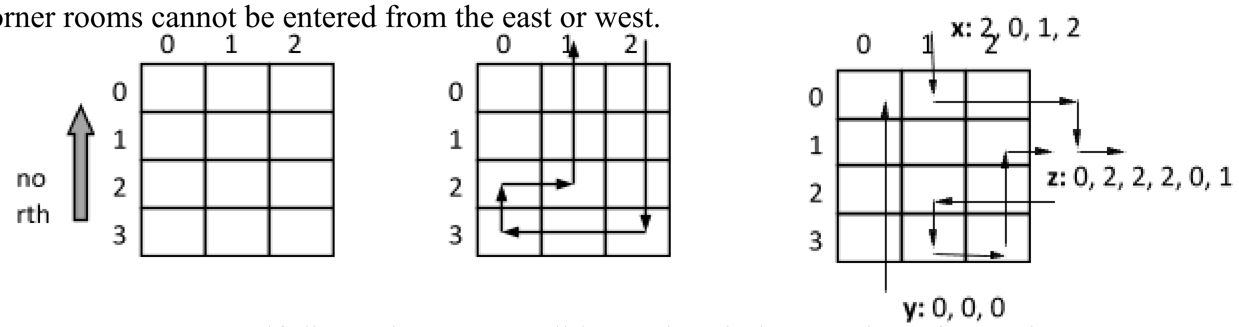
```
4
1.0 1.0
3.2 5.6
1000.53 32.65
11720.0 1.0
```

Resulting Outputs

```
1392.03 117537.31
25.36 205690.30
40.02 3835.56
16314541.69 10.00
```

6- Museum Tour
Input File: MuseumTourIn.txt

A one story rectangular building, houses a museum. Its floor plan consists of a grid of **r** by **c** equally sized rooms, as show below for **r** = 4 and **c** = 3. Interior passageways connect adjacent rooms. Rooms on the perimeter of the building also have doors on their exterior walls, however the corner rooms cannot be entered from the east or west.



Curator Jane composes self directed tours. A valid tour description consists of a starting room at which to enter the museum, and a sequence of **n** integer instructions to complete the tour *and* exit the museum. Each of these **n** instructions specify how tour members should exit the room they are in, relative to the direction they entered it: 0: proceed straight; 1: turn right; 2: turn left. They are never told to go backward. For example, if the (row, column) of a tour's beginning room was (0, 2) and the **n** = 10 instructions were: 0 0 0 1 0 1 1 2 0 0 the tour would proceed as shown in the center floor plan above.

A tour description is *not* valid (see above right floor plan) if the tour exits the museum before it ends (tour **x**); or ends without exiting the museum (tour **y**); or passes through the same room twice (tour **z**) regardless of how it would have ended if it continued beyond that room.

Inputs:
The first line contains a positive integer indicating how many tours to consider. This will be followed two lines of input per tour. The first of these lines will contain five integers. The first two integers will represent the number of rows, **r**, followed by the number of columns, **c**, in the museum's room grid. The third and four integers represent the row and column number of the starting room. The fifth integer will be the number of instructions, **n**, in the tour. The second line will be the sequence of **n** integer tour instructions, as described above. Multiple inputs on a line will be separated by a space.

Outputs:
For each tour there will be one line of output. If the tour is valid, the line will contain the grid location of each room visited on the tour in the order they are visited, formatted as: row,column with each room's location separated by a space. Otherwise the line will contain one integer whose value is 1, 2, or 3: 1 if the tour exited the building before it was completed; 2 if the tour

was completed without exiting the building; 3 if a tour would visit the same room(s) twice regardless of how it would have ended if it continued beyond that room.

(sample inputs and resulting outputs on next page)

Sample Inputs

```
9
4 3 0 2 10
0 0 0 1 0 1 1 2 0 0
4 3 0 1 4
2 0 1 2
4 3 3 0 3
0 0 0
4 3 2 2 6
0 2 2 2 0 1
3 3 0 2 10
0 0 0 1 0 1 1 2 0 0
3 3 0 0 4
2 0 1 2
3 3 2 0 3
0 0 0
4 3 3 2 6
0 2 2 2 0 1
3 3 0 2 8
0 0 1 0 1 0 1 0 0
```

Resulting Outputs

```
0,2 1,2 2,2 3,2 3,1 3,0 2,0 2,1 1,1 0,1
1
2
3
1
0,0 0,1 0,2 1,2
2,0 1,0 0,0
3
3
```

7- Tennis Anyone?
Input File: TennisAnyoneIn.txt

A group of **n** Martians and **n** Vulcans have entered a tennis tournament in which teams of two, comprised of one Martian and one Vulcan, will play each other. To decide on the team pairings, each Martian is assigned a unique random player number from 1 to **n**, and each Vulcan is assigned a unique random player number from 1 to **n**. Then each Martian and Vulcan composes a preferred partner ranking list, by listing the players' numbers they would like to be paired with in sequential preference order beginning with their most preferred partner, and ending with their least preferred partner. Then the Martians line up in numerical order, and the partner selection process is conducted as follows:

While the line is not empty, perform this process:
The Martian at the front of the line selects its next highest ranked Vulcan who has not previously "rejected" it. If that Vulcan is not currently paired with another Martian, it is paired with the Martian that selected it, and the Martian leaves the line. If that Vulcan is already paired with another Martian who it ranked lower than the Martian that just selected it, the Vulcan happily "rejects" its current partner who returns to the end of the line, the Vulcan is paired with the Martian that just selected it, and that Martian leaves the line. Otherwise, the Martian at the front of the line repeats this process until it leaves the line.

For example, if there were three (**n** = 3) Martians and three Vulcans whose partner ranking lists were as shown below, which gives Martian 3's first choice for a partner Vulcan 2 and Vulcan 1's first choice for a partner Martian 2:

	preferences				**preferences**		
	1st	**2nd**	**3rd**		**1st**	**2nd**	**3rd**
Martian1's ranking list□	2	1	3	Volcan1's ranking list□	2	3	1
Martian2's ranking list□	2	3	1	Volcan2's ranking list□	3	1	2
Martian3's ranking list□	3	2	1	Volcan3's ranking list□	2	3	1

then the (Martian, Vulcan) pairings produced by the process would be: (1, 1) , (2,3) and (3, 2). Your task is to determine the pairings given the ranking lists of the n Martians and Vulcans.

Inputs:
The first line contains a positive integer indicating how many tournaments there will be. This is followed by one group of inputs per tournament. The first line in each group contains one integer that represents the number of Martians and the number of Vulcans in the tournament, **n**. This is followed by two groupings of **n** lines that represent the Martians' and then the Vulcans' preferred partner ranking lists in player number order. The first line in these groupings represents player 1's preferred partner ranking list, the second line represents player 2 preferred partner list, etc. All inputs on these lines are separated by a space.

(continued on next page)

Outputs:
For each tournament, the program will generate a single line of output that represents the pairings produced by the process in Martian player number order, with each pairing formatted as (Martian's number, Vulcan's number).

Sample Inputs

```
2
3
2 1 3
2 3 1
3 2 1
2 3 1
3 1 2
2 3 1
4
1 2 3 4
1 4 3 2
2 1 3 4
4 2 3 1
4 3 1 2
2 4 1 3
4 1 3 2
3 2 1 4
```

Resulting Outputs

```
(1,1)(2,3)(3,2)
(1,3)(2,4)(3,1)(4,2)
```