Combination
Input File: CombinationIn.txt

Luck Smith is a locksmith who needs your help with his new design for a digital lock. Each number in a lock's combination is entered into its own cell of an n x n square grid of numbers (left side of the figure below for n = 3), with the range of the numbers being 0 to n – 1 inclusive. Adjacent to each cell around the perimeter of the lock is a number also in the range 0 to n – 1 inclusive, as shown on the center of the figure for n = 3. The lock will open when a number is placed in each cell such that the sum of it and the numbers immediately above, below, left and right of it Mod n is zero, as shown on the right side of the figure.

```
                0 1 1 1 0         0 1 1 1 0
   0 0 0        1 0 0 0 2         1 1 0 1 2
   0 0 0        1 0 0 0 2         1 0 0 2 2
   0 0 0        1 0 0 0 2         1 1 1 1 2
                0 0 0 0 0         0 0 0 0 0
```

Number Positions for a 3 x 3 Lock | **A 3 x 3 Lock** | **A Combination for the Lock**

Your task is to determine if there is a valid combination for each of Luck Smith's new locks, and determine a combination that will open the lock for each lock that has a combination. Note: there may be more than one combination for a lock.

Inputs

The first line of input contains the number of locks Luck Smith has produced. For each of these locks there will be five lines of input. The first line of input will be the number of cells in each of the lock's rows and columns. This will be followed by four lines containing n numbers that are the perimeter values adjacent to the top, bottom, left, and right sides of the lock's cells respectively. The ordering of these perimeter values on each input line are as shown in the second set of inputs/outputs given below.

Outputs

For locks that do not have a combination, there will be one line of output containing the word false. For locks that do have a combination, the first line of output will contain the word true, followed by n + 2 lines of output that show a valid combination for the lock and the perimeter values formatted exactly as shown on the right side of the figure above (with one space between each of the numbers on a line). There will be a blank line separating each lock's output.

(sample inputs and outputs are on the next page)

Sample Inputs

```
4
3
1 1 1
0 0 0
1 1 1
2 2 2
6
1 2 3 1 5 1
1 1 2 1 2 4
2 1 2 1 4 1
3 2 3 2 5 5
4
1 1 1 1
0 1 1 0
1 1 1 1
2 2 0 2
2
1 1
0 0
1 1
0 0
```

Sample Outputs

```
true
0 1 1 1 0
1 1 0 1 2
1 0 0 2 2
1 1 1 1 2
0 0 0 0 0

true
0 1 2 3 1 5 1 0
2 4 1 1 3 3 4 3
1 4 4 4 4 3 1 2
2 5 5 5 4 1 2 3
1 2 5 0 4 2 5 2
4 5 0 4 2 0 1 5
1 1 4 0 2 1 1 5
0 1 1 2 1 2 4 0

false

true
0 1 1 0
1 0 1 0
1 1 0 0
0 0 0 0
```

Hatching Eggs
Input File: HatchingIn.txt

Ryan has hired a chicken named Cathy to work in his egg hatching factory. It's Cathy's job to sit in her nest and hatch eggs, which takes exactly thirteen minutes per egg. Your task is to determine how many hours and minutes it will take Cathy to hatch all of the eggs Ryan has purchased.

Inputs:
There will be one line of input that contains the number of eggs Ryan has purchased.

Output:
There will be one line of output that contains the number of hours and minutes it will take Cathy to hatch all of the eggs Ryan has purchased, annotated exactly as shown below.

Sample input
```
125
```

Sample output
```
Cathy will be hatching eggs for 27 hours and 5 minutes
```

Latin
Input File: LatinIn.txt

Knowing that Thomas Jefferson wrote notes to his boyhood friends in Pig Latin, Nora has decided to translate one of his other documents, the Declaration of Independence, into that language. Since it is a long document she has asked you to automate the translation process, which as two rules:

1. For words that begin with a vowel, add the three letters "way" to the end of the word (e.g., the word: one translates to: oneway).
2. For words that do *not* begin with a vowel move the first letter of the word to the end of the word, make it a lower case letter, and then add the two letters: "ay" to the end of the modified word (e.g., the word: human translates to: umanhay)

When using either rule, the case (upper or lower) of the first letter of the translated version of the word will always be the same as that used in the English version of the word. Thus, the word: Events translates to: Eventsway and the word: Course translates to: Oursecay.

Your task is to write a program that accepts the text of an English language document and translates it, line-by-line, into the Pig Latin version of the document.

Inputs
The first line of input contains the number of lines in the document, n. This will be followed by n lines of input, which are the text of the document.

Outputs
There will be n lines of output that are the line-by-line translation of the input document, as shown below (note the use of capitalization).

Sample Inputs
```
4
When in the Course
of human Events
it becomes necessary
for one people
```

Sample Outputs
```
Henway inway hetay Oursecay
ofway umanhay Eventsway
itway ecomesbay ecessarynay
orfay oneway eoplepay
```

Lucky Numbers
Input File: LuckyIn.txt

Lucky Lazarus and his friends are conducting a lottery in which the winning ticket number is the `n`th integer in a number sequence Lazarus has conceived. As shown below, the sequence generation process begins by placing the first 10,000 non-negative odd integers in ascending order. Since the second number in this sequence is 3, Lucky eliminates every 3rd number in this sequence to form a new sequence. Since the third number in this new sequence is 7, every 7th number in this new sequence is eliminated from it to form a new sequence. Since the forth number in this new sequence is 9, every 9th number in this new sequence would be eliminated from it to form a new sequence. This process is repeated until the `n`th number in the sequence is generated, which is the winning number.

The sequence generating process is shown below for `n` = 4, which yields a winning number of 9.

1 3 5 7 9 11 13 15 17 19 21…	The sequence of positive odd integers in ascending order
1 3 7 9 13 15 19 21…	Every third number removed from the previous sequence
1 3 7 9 13 15 21…	Every seventh number removed from the previous sequence

Inputs
The first line of input contains the number of Lottery games to consider. For each of these games, there will be one line of input that specifies sequence term, `n`, of the winning ticket number.

Outputs
There will be one output per Lottery game that contains the winning ticket number for that game.

Sample Inputs
```
5
1
26
22
5
2066
```

Sample Outputs
```
1
115
93
13
19993
```

Maggie's Money
Input File: MaggieIn.txt

Magnificent Maggie can purchase lots of clothing using very little money. To accomplish this she vacations in places that do not charge sales tax, and focuses on stores that are running sales. Her favorite store is running a sale in which the discount rate, applied to the total cost of the purchased items, is a function of the total cost of the items as shown in the below table. Your task is to determine the total of her shopping spree purchase after the discount adjustment.

Total Purchase Price	Discount Rate
greater than $100.00	35%
greater than $75.00	20%
greater than $50.00	10%
greater than $25.00	5 %

Inputs
The first line of input contains the number of shopping sprees to consider. For each of these sprees, there will be one line of input that specifies the total of Maggie's purchases while on that spree.

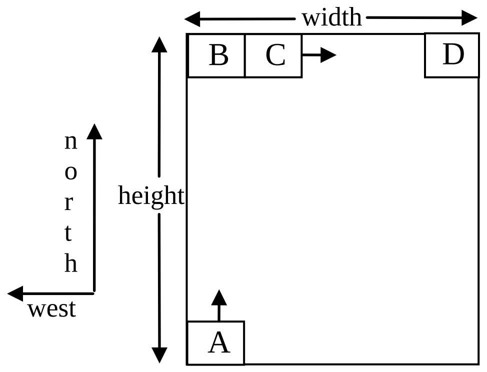
Outputs
There will be one output per shopping spree that contains the discounted cost of the items purchased for that shopping spree formatted as US currency rounded to the nearest penny, as shown below.

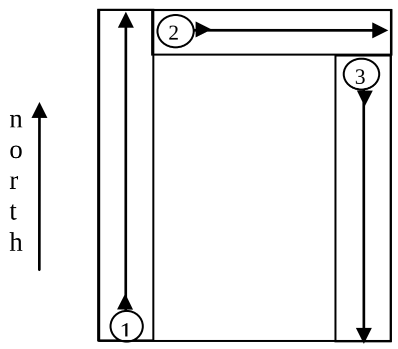
Sample Inputs
```
7
25.00
25.01
60.00
70.04
80.00
100.00
6000.10
```

Sample Outputs
```
$25.00
$23.76
$54.00
$63.04
$64.00
$80.00
$3,900.07
```

Mower
Input File: MowerIn.txt

Logan cuts rectangular lawns, whose height is always larger than its width, using a lawn mower whose blades are arranged in such a way that it cuts a square patch of lawn. As shown on the left side of the below figure, he begins by placing the left side of the mower on the west edge of the lawn, and the rear edge of the mower on the south edge of the lawn (A). Then he proceeds straight ahead until the front of the mower reaches the uncut edge of the lawn (B). If there is more lawn to cut, he makes a 90 degree right turn pivoting on the mower's back right wheel (C), and then proceeds straight ahead until the front of the mower reaches the uncut edge of the lawn (D). He repeats this process as shown on the right side of the below figure, which shows the first three cuts, until the entire law is cut.





Your task is to determine how many seconds it takes him to cut the lawn, given the width of the lawn mower's square cut, the height and width of the lawn, his walking speed, and the time it takes him to make a 90 degree turn. Do *not* include the time it takes to place the mower on the lawn at its initial position (i.e., position A shown on the left side of the above figure).

Inputs:
The first line of input contains the number of lawns to cut. This will be followed by one line of input per lawn that contains five doubles: the width of the lawn mower's square cut in feet, followed by the height and width of the lawn in feet, his walking speed in feet per second, and the time it takes him to make a 90 degree turn in seconds. The inputs on one line will be separated by a space.

Outputs:
There will be one line of output per lawn containing the number of seconds required to cut that lawn, rounded to the nearest hundredth of a second.

Sample Input
```
6
5.0 185.0 5.0 3.0 2.0
5.0 185.0 10.0 3.0 1.0
5.0 185.0 15.0 3.0 1.0
5.0 14.0 11.0 3.0 1.0
5.0 18.0 11.0 3.0 1.0
2.0 100.0 40.0 3.0 1.0
```

Sample Output
```
60.00
120.33
180.67
8.67
12.33
624.67
```

Numbers
Input File: NumbersIn.txt

Paranoid Paul has invented a new scheme for encrypting 32 bit integer data that he transmits to his friend Skyler. The scheme begins with the selection of an integer encryption key, `n`, in the range $1 \leq n \leq 4$. Then adjacent bits of the integer to be encrypted are grouped into 2^n bit groupings, and the position of each adjacent pair of groupings is swapped to form the encrypted number. For example, the unencrypted and encrypted versions of the integer 16,711,850 using an encryption key of n = 3, is shown below.

```
00000000111111110000000010101010  Unencrypted
11111111000000001010101000000000  Encrypted
```

After the encryption, the encryption key `n` and the encrypted number are transmitted to Skyler for decryption. Your task is to produce the encrypted version of an integer given the integer and the encryption key, `n`.

Inputs
The first line of input contains the number integers to be encrypted. This is followed by one line of input per number to be encrypted, that contains the value of the encryption key `n` (in the range $1 \leq n \leq 4$) to be used in the encryption, followed by the integer to be encrypted.

Outputs
There will be one line of output per encrypted number containing three integers: the number to be encrypted, followed by the value of the key (`n`) used in the encryption, followed by the encrypted version of the number. All outputs will be separated by one space.

Sample Inputs
```
6
1 195
3 16711850
2 15
3 195
4 15
1 -1
```

Sample Outputs
```
195 1 60
16711850 3 -16733696
15 2 240
195 3 49920
15 4 983040
-1 1 -1
```