1. Base Three

Program Name: Base3.java   Input File: base3.dat

Geek High School's local math club, Mu Alpha Theta, has decided to use a code for numbers used by their members. The code they have adopted will use M, A, and O for their "digits". The decimal counterparts of these digits are 1, -1, and 0 respectively. Each place value of a digit in a number is three times larger than the digit to its right with the rightmost place value being a one. For example, the Geek number MAO would be equivalent to 6 since 9*1 + 3*(-1) + 1*0 = 6.

Input
Each of line of input will contain a single Geek number.

Output
For each Geek number, you will print its base 10 equivalent on a single line.

Example Input File
```
MAO
MMAAOO
MAOMAO
OMOAO
MAMA
```

Example Output to Screen
```
6
288
168
24
20
```

2. Duplicate Vowels

Program Name: Duplicates.java   Input File: duplicates.dat

There are many words that contain all of the vowels, A, E, I, O, and U. There are also many words that contain all of the vowels and the vowels appear in alphabetical order when duplicate vowels are ignored.

For example, `S`**`A`**`CRIL`**`E`**`G`**`IOU`**`S` is a word that contains all five vowels in alphabetical order when the vowel `I` that appears after the `R` is ignored.

You are to write a program that will determine if a given word contains all five vowels and the vowels appear in alphabetical order, ignoring any duplicates as described above as necessary.

Input
The first line of input will contain a single integer `n` that indicates the number of words to follow. Each of the next `n` lines will contain a single word in all caps.

Output
For each word, you will print `YES` if all five vowels appear in alphabetical order according to the criteria above or `NO` if they do not.

Example Input File
```
3
SACRILEGIOUS
AUTOMOBILE
AUTOECIOUSLY
```

Example Output to Screen
```
YES
NO
YES
```

3. Exponents

Program Name: Exponents.java   Input File: exponents.dat

Many codes are developed that use prime numbers and powers of prime numbers. For this problem, you will be given a prime number and a second number which is a power of that prime number. You are to write a program to find what power of the first number that the second number is.

Input

Each line of input will contain two integers separated by a space. The first integer is a prime number `n`, where $1 < n < 20$, and the second integer is n^p, where $0 < p < 15$.

Output

For each line of input, you will print one line of output that contains the value of `p`, the power to which `n` is raised.

Example Input File

```
2 16
3 243
11 3138428376721
```

Example Output to Screen

```
4
5
12
```

4. Four of a Kind

Program Name: FourKind.java   Input File: fourkind.dat

There are many games that are played with 5 standard six-sided dice with numbers one through six on a side. In a poker type game, one of the hands a player can have is four of a kind. A four of a kind hand consists of four dice that have the same number and the remaining die has a different number. For example, `3 3 3 3 5` is a four of a kind hand; `3 3 3 3 3` and `2 3 3 4 3` are not four of a kind hands. If two or more hands have four of a kind, ties are broken first by the number on the four matching dice, with one being the lowest and six being the highest. If four dice match in two or more hands, the tie is broken by the fifth die, again with one being the lowest and six being the highest. If the tie is not broken by one of these two methods, a tie exists.

For the purpose of this program, you will be given the outcomes after each of three players has rolled their dice. You are to determine if any of the players has a four of a kind hand and, if more than one player has a four of a kind hand, which player or players have the highest four of a kind hand.

Input

The first line of input will contain a single integer `n` that indicates the number of games to be played. For each game, there will be three lines with each line containing:

- The player's first name followed by a space.
- Five integers indicating the numbers on each of the five dice, each separated by a space.

Output

For each game

- If there is exactly one person with a four of a kind hand, you will print the name of that person.
- If there is more than one person with a four of a kind hand, you will print the name of the person with the highest four of a kind. For example, a hand of `5 5 5 5 2` would beat a person with a hand of `4 4 4 4 6`.
- If there is more than one person with the same four of a kind hand, you will check the fifth die to see which hand is the highest and print the name of the person with the highest hand.
- If there are two or more people with the same hand, then there is a tie and you will print, on a single line, the word `TIE` followed by the names of the players who were tied in alphabetical order and separated by a space.
- If no hand contains four of a kind, you will print `NO WINNER`.

Example Input File

```
4
MARY 4 4 4 4 3
JOAN 4 4 4 2 3
ART 3 3 3 2 1
RON 4 4 4 4 1
ALEX 4 3 4 4 4
CHEZ 2 6 6 6 3
AMY 4 4 4 4 5
REX 4 5 4 4 4
RICK 2 2 4 2 3
GEORGE 5 4 5 4 1
JOHN 5 5 4 5 3
DICK 1 2 3 4 5
```

Example Output to Screen

```
MARY
ALEX
TIE AMY REX
NO WINNER
```

5. House Numbers

Program Name: House.java   Input File: house.dat

In the town of Geekville, it was decided that all numbers in a street address would be given in binary. Therefore, all residents had to buy new numbers for their house. Since it takes more metal to make a zero than it takes to make a one, the cost of the digits were sometimes different. You are to write a program that will compute the cost of the new numbers.

Input
The first line of input will contain a single integer `n` that indicates the number of house numbers to follow. Each of the following `n` lines will contain three base 10 integers separated by a single space. The first integer is the cost of a zero in pennies, the second integer is the cost of a one in pennies, and the third integer is the house number.

Output
For each line of input, you will print the binary address, a space, a dollar sign, and the cost of the binary address in dollars and cents. If the cost is less than one dollar, one leading zero shall be printed.

Example Input File
```
4
35 25 111
45 40 2104
11 11 32
33 32 1515
```

Example Output to Screen
```
1101111 $1.85
100000111000 $5.20
100000 $0.66
10111101011 $3.55
```

6. Let’s Make a Deal

Program Name: LetsDeal.java   Input File: letsdeal.dat

In the TV show, Let’s Make a Deal, the probability that a person will choose the door with the prize with the highest value behind it is 1/3. However, the rules have been tweaked so the player is allowed to look behind one door, decide if he wants that prize or if he wants to reject that prize. If he chooses to reject that prize, he is allowed to look behind a second door and decide if he wants that prize or if he wants the prize behind the unopened third door. He is not allowed to open the third door before making his decision on which door he will choose.

Being a smart man, the player decides that he can improve his chances of winning to ½ if he uses the following strategy: He will randomly choose a door to open, look at the value, and then reject what is behind that door. He will then select one of the two remaining doors. If the value behind this door is greater than the value behind the first door he opened, he will choose the item behind that door. Otherwise, he will choose the item behind the unopened door.

Input

The first line contains a single integer `n` that indicates the number of games to be played. Each of the next `n` lines will contain six distinct integers in the format: `v1 v2 v3 c1 c2 c3`. The integers `v1`, `v2`, and `v3` indicate the values of the items behinds doors 1, 2, and 3 respectively. The integer `c1` denotes the first door selected, the integer `c2` denotes the second door selected and the integer `c3` indicates the remaining door that may or may not be selected. **Note:** All three integers are listed as input even if he did not open the last door.

Output

Print the number of the door the player chose followed by the word `WON` if he won the item worth the most or the word `LOST` if he won one of the other two items.

Example Input File

```
2
100 200 300 2 1 3
400 200 10 3 2 1
```

Example Output to Screen

```
3 WON
2 LOST
```

7. Lockers

Program Name: Lockers.java   Input File: lockers.dat

James left Lloyd's homework in his locker and, being the obnoxious friend that he is, James refuses to tell Lloyd the locker number. Instead, James wants to play a game with him. James will have Lloyd stand in the hall where the locker is located and have Lloyd read a locker number to him. James will help Lloyd by telling him if he is getting closer or further away from his locker until Lloyd eventually finds the correct locker.

Since James is lazy, he wants you to write a program for Lloyd to use on his laptop so James does not have to go with Lloyd in this endeavor. He can also use the program for other unsuspecting classmates who make the mistake of giving him their homework.

Input

The first line of input will contain a single integer `n` that indicates the number of lockers to be found. Each of the next `n` lines will contain the following:

- The number of the first locker in the hall followed by a space.
- The number of the last locker in the hall followed by a space.
- The number of the target locker (the locker that contains the homework) followed by a space.
- The first locker number guessed followed by a space.
- The remaining locker numbers guessed in the order that they were guessed separated by a space.

Note: All lockers are located on one side of the hall. The locker numbers are in sequential order. All locker numbers in the input file will be in the range given.

Output

If the target locker is the first locker guessed, print the locker number, a space, and `FOUND ON FIRST TRY`. Otherwise, for each locker number guessed, except the first, print the locker number followed by a space and one of the following:

- `COLDER` if he has moved further away from the target locker,
- `WARMER` if he has moved closer to the target locker,
- `SAME` if he is the same distance from the target locker as the previous guess
- `FOUND` when he finds the target locker.

Print exactly one blank line before continuing to the next locker to be found.

Example Input File

```
4
500 1000 859 575 600 650 732 840 950 850 900 875 859
100 400 150 300 350 200 175 125 400 150
1000 2000 1575 1428 1300 1400 1450 1700 1600 1500 1575
500 700 650 650
```

Example Output to Screen

```
600 WARMER
650 WARMER
732 WARMER
840 WARMER
950 COLDER
850 WARMER
900 COLDER
875 WARMER
859 FOUND

350 COLDER
200 WARMER
175 WARMER
125 SAME
400 COLDER
150 FOUND

1300 COLDER
1400 WARMER
1450 WARMER
1700 SAME
1600 WARMER
1500 COLDER
1575 FOUND

650 FOUND ON FIRST TRY
```

8. Lucky Number

Program Name: Lucky.java   Input File: none

Xavier needs to make a stencil to use to paint his lucky number on his book covers. You are to write a program that will create this stencil for him.

Input
There is no input for this problem.

Output
Print this stencil exactly as it appears below.

Example Input File
There is no input for this problem.

Example Output to Screen
```
************************
*                      *
********************  **
*******************  ***
******************  ****
*****************  *****
****************  ******
***************  *******
**************  ********
*************  *********
************  **********
***********  ***********
**********  ************
*********  *************
********  **************
*******  ***************
******  ****************
*****  *****************
****  ******************
***  *******************
**  ********************
*  *********************
************************
```

9. Money: Brother Can You Spare a Hammer?

Program Name: Money.java   Input File: money.dat

Henry Putter lives in a world different than ours. The money is very different. It consists of bronze "soups", silver "hammers", and golden "frigates". One silver hammer is worth 7 bronze soups. One golden frigate is worth 11 silver hammers or 77 bronze soups. Henry wants to know how many combinations of coins exist for various amounts.

For example:

- There is only one way to combine the coins to have 5 soups worth of coins, the 5 soups.
- There are 2 ways to have 13 soups worth of coins:
  - 13 soups
  - 1 hammer and 6 soups. (Recall 1 hammer is worth 7 soups.)
- There are 3 ways to have 20 soups worth of coins:
  - 20 soups
  - 1 hammer and 13 soups
  - 2 hammers and 6 soups
- There are 13 ways to have 77 soups worth of coins:
  - 77 soups
  - 11 hammers
  - 1 frigate
  - 10 distinct combinations of soups and hammers. (10 hammers and 7 soups, 9 hammers and 14 soups, 8 hammers and 21 soups, and so forth.)

Input

- The first line will contain a single integer `n` that indicates the number of data sets that follow.
- Each data set will consist of single integer on a single line. This is the target number of soups.
- Each integer will be greater than 0 and less than 10,000

Output

For each data set print out the number of combinations of coins that exist that equal the target number of soups.

Example Input File

```
4
5
13
77
200
```

Example Output to Screen

```
1
2
13
54
```

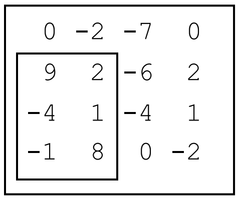
10. Realtor World

Program Name: Realtor.java Input File: realtor.dat

Scott works for a large realty firm. His boss is requiring each realtor in his firm to select a portion of a nearby housing area to contact the residents about selling their homes. The housing area is rectangular in shape and, ignoring the streets, there are `m` houses across and `n` houses down. Each realtor is to select a sub-rectangle of contiguous houses within the housing area. The realtor who is most successful will be rewarded with an all-expense paid vacation for a family of 4.

Scott, one of the realtors, has decided that he needs a program to help him select his sub-rectangle. A sub-rectangle must be at least a 1x1 rectangle. After some investigation, he has developed a system that will give each house a value between -10 and 10, inclusive, based on how long a house has been owned, how long the owners had owned their previous house, value of the house, and several other criteria.

You are to write a program for Scott that will determine the sub-rectangle with greatest value. For example, given the housing area to the right, Scott has given the values of each house. The sub-rectangle containing `9, 2, -4, 1, -1, 8` sums to 15 and that is the largest value for any possible sub-rectangle.



```
 0 -2 -7  0
 9  2 -6  2
-4  1 -4  1
-1  8  0 -2
```

Input

The first line of input will contain a single integer `n` that indicates the number of housing areas to follow. For each housing area, there will be the following:

- One line with two positive integers `r` and `c`, separated by a space, to denote the number of rows and columns in the housing development.
- `r` lines, each with `c` integers separated by a space and in the range [-10,10].

Output

For each housing area, you will print one line with the value of the largest valued sub-rectangle.

Note: There will always be a unique largest valued sub-rectangle.

Example Input File

```
2
4 4
0 -2 -7 0
9 2 -6 2
-4 1 -4 1
-1 8 0 -2
4 6
3 3 -3 -3 4 -4
2 8 3 8 9 10
-4 -4 3 7 -5 3
6 2 -2 4 -6 8
```

Example Output to Screen

```
15
52
```

11. Target Practice

Program Name: Target.java   Input File: target.dat

Don and his fellow police officers are having target practice. They have decided to make a contest out of it, and have come up with the following rules:

- Each person fires 5 shots at a target.
- The center of a target and the position of a shot are each recorded as `(x,y)` coordinates.
- The distance of each of the 5 shots from the center of the target can be determined by using the following general formula for the distance between two points:

  $$\sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2}$$
- The scoring system they have decided on is to take each officer's best 3 shots. An officer's best 3 shots are his 3 shots that are closest to the center of the target.
- The score for a given target and a given officer is determined by adding the distances for those 3 shots together and then rounding the total to the nearest integer.
- The officer with the lowest score for a target wins on that target.
- For each officer the same 5 shots will be considered for each of the different targets.

Given the `x` and `y` coordinates for each officer's 5 shots and the center of various targets determine which officer wins on a target and what their score is. The given data will never result in a tie.

Input

- The first line will contain two integers `m` and `n`. The integer `m` indicates the number of officers who have shot at the targets. The integer `n` indicates the number of different targets.
- The next `m` lines contain the data for the officers.
  - Each line of data for the officers will have the following form:
    NAME (x_1,y_1) (x_2,y_2) (x_3,y_3) (x_4,y_4) (x_5,y_5)
  - The officer's name will consist of uppercase letters only and will be followed by a single space.
  - Following the name are the locations of the 5 shots in the form `(x,y)` and separated by a single space.
  - All `x` and `y` coordinates will be integers greater than -100 and less than 100.
- The next `n` lines will be the location of the center of the targets.
  - Each line will be of the form `(x,y)` which are the coordinates for the center of that target.
  - All `x` and `y` coordinates will be integers greater than -100 and less than 100.

Output

For each target print out `TARGET <N> <NAME> <SCORE>.`

- `<N>` is the number of the target which corresponds to its order in the input data set, starting at 1.
- `<NAME>` is the name of the officer who had the lowest (best) score on that target given his shots.
- `<SCORE>` is the score for the winning officer for that target rounded to the nearest integer.

Example Input File

```
3 2
PYLE (3,4) (5,4) (0,0) (4,1) (10,11)
ANDY (1,5) (-1,-2) (3,3) (-4,4) (0,1)
DON (0,2) (-2,0) (1,0) (2,3) (-2,1)
(0,0)
(15,-10)
```

Example Output to Screen

```
TARGET 1 DON 5
TARGET 2 PYLE 51
```

12. Treasure Hunt

Program Name: Treasure.java   Input File: treasure.dat

Jack has found a treasure map but only has a limited amount of time to pick up the treasure. Write a program that takes a treasure map and determines the maximum treasure that can be gathered in a given number of moves.

- The map will consist of integers between 0 and 9 inclusive. These numbers indicate the amount of treasure at the corresponding location.
- Jack can take a given number of steps. Jack can move North, South, East, or West only. Each step moves Jack one spot.
- Jack's initial position will be shown with an asterisk (*) on the edge of the map. It is possible Jack starts on a corner and so he has to take 2 steps to get onto the portion of the map that can contain treasure.
- Jack does not need to end up where he started. He can end anywhere on the map or on the boundary of the map.
- When Jack enters a location he picks up the treasure there. If he steps into that location again there isn't any more treasure at that location.

For a given map, starting location, and number of steps print out the maximum treasure Jack can collect.

Input

- The first line will contain a single integer `n` that indicates the number of data sets that follow.
- The first line in each data set will be an integer `steps` indicating the number of steps Jack can take.
- The second line of a data set will be two integers, `row` and `column`. These indicate the number of rows and columns in the treasure map not including the boundaries. All maps will be rectangular.
- The next `row + 2` lines will be the treasure map and the boundaries of the map. Each line will contain `column + 2` characters. The boundaries of the map will be indicated with periods (.), except for Jack's initial position which will be marked with an asterisk (*).

Output

For each data set print out maximum treasure Jack can collect on the map based given his starting location and the maximum number of steps

Example Input File

```
2
5
3 6
........
.000900.
*100000.
.161101.
........
8
5 7
........*
.9000112.
.0102045.
.0701020.
.8412302.
.3332221.
.........
```

Example Output to Screen

```
10
18
```