1. Boxes

Program Name: Boxes.java **Input File: boxes.dat**

A popular Russian item is a set of nesting dolls, also known as Matryoshka dolls. These are a series of dolls that fit one inside each other, from smallest to biggest. As you are packing to move into your college dorm, you wish to apply this concept to fit a number of cardboard boxes into each other to make them easier to carry.

Given a series of box dimensions, write a program to determine how many boxes are able to fit inside each other, taking each box and placing it (after possibly rotating it) inside the next one.

Input

- The first line will contain a single integer `n` that indicates the number of data sets to follow.
- Each data set will consist of:
  - A line containing an integer `b` that indicates the number of boxes, 2 <= `b` <= 10.
  - A line containing a space-separated list of `b` box dimensions in the format "`LxWxH`", where `L` represents the length of the box in feet, `W` represents the width of the box in feet, and `H` represents the height of the box in feet, 1 <= `L`,`W`,`H` <= 20. Note that although these are the given dimensions, a box can be rotated such that its dimensions change (e.g., a 1x2x3 box can be rotated to be a 1x3x2, 2x1x3, 2x3x1, 3x2x1, or 3x1x2 box).

Output

For each data set in the input, output a single line "`X`", where `X` is the number of boxes that are able to fit into its next box in the given sequence of boxes, stopping once a box does not fit into the next box in the sequence. In order to fit, each of the box's dimensions must be strictly smaller than the dimensions of the box it is placed in.

Example Input File

```
4
2
1x2x3 2x3x4
2
3x1x2 2x3x4
2
1x2x3 2x3x3
3
1x2x3 2x3x4 3x4x4
```

Example Output to Screen

```
1
1
0
1
```

2. Encryption

Program Name: Encryption.java Input File: encryption.dat

The Texas Gnome Eatery does most of its sales with credit cards. In an effort to secure their data during transmission to their financial headquarters, they encrypt the 16 digit credit card numbers by using this algorithm:

- For each digit `d`, substitute `d = (d + 7) mod 10`.
- Divide the new digits into four groups of four digits:
    - For each group of four digits, interchange the first and third digits.
    - For each group of four digits, interchange the second and fourth digits.

You are to write a program that will encrypt the 16 digit credit card numbers as described above and print the resulting 16 digit number.

Input

The first line of input will contain a single integer `n` that indicates the number of credit card numbers to follow. Each of the following `n` lines will contain a 16 digit credit card number.

Output

For each credit card number input, print the encrypted number.

Example Input File

```
3
1234567890123456
4536278290123476
7685438920456431
```

Example Output to Screen

```
0189452389672301
0312599489674301
5243561012970831
```

3. FizzBang!!!

Program Name: Fizz.java **Input File: fizz.dat**

Fizz Bang is a children's counting game. Beginning with the starting value, numbers are counted out loud with the following exceptions:

- A number that is a multiple of 3 and 5 is replaced with the phrase "`FizzBang`".
- A number that is a multiple of 3 but not 5 is replaced with the phrase "`Fizz`".
- A number that is a multiple of 5 but not 3 is replaced with the phrase "`Bang`".

Input

- The first line will contain a single integer `n` that indicates the number of data sets to follow.
- Each data set will consist of:
  - A single line of the form `start end` where `start` is the starting number in the count and `end` is the last number in the count.
  - `start` and `end` will both be greater than 0.
  - `start` will be less than or equal to `end`.
  - `end` will be less than or equal to 2,000,000,000.

Output

For each data set print out the phrase "`Data Set m` " on a line by itself, where `m` is the number of the data set. Then print the values from `start` to `end`, one value per line, following the rules of the FizzBang counting game,

Example Input File

```
4
5 5
6 10
10 15
20 30
```

Example Output to Screen (see next page)

3. FizzBang!!! (cont.)

Example Output to Screen

```
Data Set 1
Bang
Data Set 2
Fizz
7
8
Fizz
Bang
Data Set 3
Bang
11
Fizz
13
14
FizzBang
Data Set 4
Bang
Fizz
22
23
Fizz
Bang
26
Fizz
28
29
FizzBang
```

4. Intersection

Program Name: Intersection.java Input File: intersection.dat

Geoff is working a project that has several sets of data. To complete his project, he must analyze the elements that are common to all sets in his current project. Your job is to write a program for Geoff that will find all elements that appear in all of the sets in a particular project. All data sets contain only positive integers and an integer may appear more than once in the data set.

Input

- The first line of input will contain a single integer `p` that indicates the number of projects to follow.
- Each project will consist of:
  - A line containing the number `n` that indicates the number of sets of data for that project.
  - Each of the next `n` lines will contain a finite number of integers, each separated by a single space, which is the data for that particular set of the project.

Note: There will be a minimum of two and a maximum of ten data sets in any project.

Output

For each project, you will print all of the elements that are common to all of the sets in that project in numerical order, on a single line, and separated by whitespace. No integer should be printed more than once even if it appears more than once in all data sets in the project.

If there is no element common to all of the sets in a project, print `NO COMMON ELEMENT FOUND`.

Example Input File

```
3
3
1 2 3 4 5 5
3 4 5 6 7 5
4 5 1 2 3 5 6 7 8 9
5
10 11 15 61 42 14 112 34
12 13 11 61 42 10 34 12 14 112
10 10 13 11 112 54 61 13 13 34
11 112 15 42 13 34 17 10
71 10 11 112 75 63 15 42 34 56 67
4
1 2
3 4 5
6 7 8 9
10 11 12 13 14
```

Example Output to Screen

```
3 4 5
10 11 34 112
NO COMMON ELEMENT FOUND
```

5. A Long Walk on Elm Street

Program Name: LongWalk.java **Input File: longwalk.dat**

Fred wants to know which houses on a given street are the farthest apart. He has compiled house addresses for various streets by scanning the yellow pages, but the addresses are not in any particular order. Write a program to determine which houses on a street are farthest apart based on street address.

Input

- The first line will contain a single integer `n` that indicates the number of data sets to follow.
- Each data set will consist of:
  - A line that contains the name of the street for that data set.
  - A line that contains an integer `m` that indicates how many addresses follow, where `m` will be greater than 1.
    - The next `m` lines will be integers representing street addresses.
    - Each address will be greater than 0 and less than one million, 0 < address < 1,000,000.
    - No address will appear more than once per data set.

Output

For each data set print out the name of the street and the street addresses, including the street name, of the two houses that are farthest apart on that street. The smaller street address shall be listed first. The output for each data set is placed on two lines as shown below.

Example Input File

```
3
Elm Street
3
1001
1210
514
El Camino Real
4
99999
12
10000
9999
Broadway
3
1
999999
123456
```

Example Output to Screen

```
The two houses farthest apart on Elm Street are
514 Elm Street and 1210 Elm Street
The two houses farthest apart on El Camino Real are
12 El Camino Real and 99999 El Camino Real
The two houses farthest apart on Broadway are
1 Broadway and 999999 Broadway
```

6. Mailboxes, Etc.

Program Name: Mail.java **Input File: mail.dat**

"I need to open as many of these mailboxes as possible," you mutter to yourself, as you eye the P.O. boxes before you. You examine the clay master key in your left hand and ponder the mailbox layout sheet in your right. You know that the clay master key will open any of the mailboxes, but will crumble after a single use. From the mailbox layout, you know which mailboxes are empty and which contain numbered keys that open other mailboxes. Write a program to find the maximum number of mailboxes you are able to open with a single use of a master key (continuously opening any unopened mailboxes you subsequently find keys to).

Input

- The first line will contain a single integer `n` that indicates the number of data sets to follow.
- Each data set will consist of:
  - A line containing an integer `b` that indicates the number of mailboxes, 1 <= `b` <= 30.
  - The next `b` lines will describe the contents of each mailbox:
    - The first line corresponds to mailbox #1, the second line corresponds to mailbox #2, etc.
    - Each contents line will consist of an integer `k` followed by a space-separated list of `k` unique ordered integers, with each integer representing a key to the mailbox with that number.

Output

For each data set in the input, output a single line "`X`", where `X` is the maximum number of mailboxes you are able to open with a single use of a master key (continuously opening any unopened mailboxes you subsequently find keys to).

Example Input File

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

Example Output to Screen

```
3
1
3
```

7. Mathabetical Order

Program Name: Mathabet.java **Input File: mathabet.dat**

Given a list of words, arrange them in ascending "mathabetical order". What is mathabetical order? Mathabetical order can be determined for a word by first substituting each letter with a number equal to its position in the alphabet (e.g., a=1, b=2, c=3, etc.) Comparisons are then made between two of the resulting numbers accordingly:

- First, a prime number is considered greater than a composite number.
- Next, if both numbers are primes or both are composites, an even number (divisible by 2) is considered greater than an odd number.
- Finally, if both numbers are equal in the two above categories, then they are compared numerically.

As an example, let's compare the words "`ag`", "`bury`", "`cry`", and "`dad`".

Word	Resulting number	Prime	Even
ag	17	Y	N
bury	2211825	N	N
cry	31825	N	N
dad	414	N	Y

Therefore the given words in ascending mathabetical order would be:

```
cry
bury
dad
ag
```

Input

- The first line will contain a single integer `w` that indicates the number of words to sort, 1 <= `w` <= 20.
- Each of the next `w` lines will contain a single word. For the purposes of this problem, a word will be a group of 1-4 lowercase letters.

Output

Output the given list of words in mathabetical order, with one word per line. Note that there will be no "ties" amongst the given words.

Example Input File

```
4
ag
bury
cry
dad
```

Example Output to Screen

```
cry
bury
dad
ag
```

8. Room for Improvement

Program Name: Room.java **Input File: room.dat**

You would like to replace the ratty carpet in your room. As you've been working part-time at a major home-improvement chain, you can purchase new carpet for $1 per square foot. Now all you have to do is calculate the size of your room.

You will be given a map of your house with `#` signs indicating the walls and an `@` sign indicating the spot where you are standing. The 'room' to be carpeted is all the area that you can access without moving through a wall, and you are guaranteed that the room will always be completely enclosed by walls (i.e., your room is not outside).

You are to write a program to determine the cost to purchase the new carpet.

Input

- The first line will contain a single integer `n` that indicates the number of data sets in the input.
- Each data set will consist of:
  - a line containing two integers `r` and `c` (each between 1 and 50, inclusive), indicating the number of rows and columns in this house map.
  - `r` lines, each containing `c` characters, making up the map.
  - Each character will be one of:
    - `#` - a wall
    - `.` (period) – empty space representing 1 square foot of floor area
    - `@` - you (standing in an empty space in your room)

Output

For each data set in the input, output a single line indicating the number of dollars you must spend to buy the carpet for your room assuming that there is no waste (i.e., you are able to buy exactly the right amount of carpet in the right shape to fit your room at the given unit cost per square foot).

Example Input File

```
2
5 5
#####
#...#
#.@.#
#...#
#####
7 19
###################
#.................#
#.################
#...###.........#..
#.@.#.###########..
#...###............
#####..............
```

Example Output to Screen

```
$9
$27
```

9. Stairs

Program Name: Stairs.java Input File: none

Write a program that will print the staircase below.

Input

No input.

Output

Print this staircase exactly as it appears below.

Example Input File

There is no input for this problem.

Example Output to Screen

```
**************
**************
******************
******************
**********************
**********************
**************************
**************************
******************************
******************************
**********************************
**********************************
**************************************
**************************************
******************************************
******************************************
**********************************************
**********************************************
**************************************************
**************************************************
```

10. Top Ten Reasons ...

Program Name: TopTen.java Input File: topten.dat

A late night television show host frequently lists his top ten reasons for a variety of events. However, he always reads the items in reverse. That is, he reads his #10 reason first and continues reading in order until finally reading his #1 reason last.

Input

The top ten reasons for competing in the Regional Computer Science competition in order from 1 to 10.

Output

You are to print, one per line, the ten reasons in order from10 to 1.

Note: In the samples below, the input/output data for reasons 6 through 10 are not shown.

Example Input File

```
1. Taking hardware home to show my parents.
2. Eating junk food for 2 days.
3. Staying overnight in a crowded hotel room.
4. Riding miles and miles in a crowded vehicle.
5. Staying up all night to play video games.
```

6. ... not shown
7. ... not shown
8. ... not shown
9. ... not shown
10. ... not shown

Example Output to Screen

10. ... not shown
9. ... not shown
8. ... not shown
7. ... not shown
6. ... not shown

```
5. Staying up all night to play video games.
4. Riding miles and miles in a crowded vehicle.
3. Staying overnight in a crowded hotel room.
2. Eating junk food for 2 days.
1. Taking hardware home to show my parents.
```

11. Vector Addition

Program Name: Vectors.java Input File: vectors.dat

A mathematician's definition of a two-dimensional vector in the Cartesian plane is an ordered pair (x,y) of real numbers. A frequent operation with vectors is addition. The vector sum, or resultant, of two or more vectors is found by adding the x-component of each vector to find the x-coordinate of the resultant and by adding the y-component of each vector to find the y-coordinate of the resultant. The resultant is then reported as an ordered pair, (x,y).

Input

The first line of input will contain a single integer `n` that indicates the number of vector addition problems to follow. Each of the following `n` lines will contain one problem for which you will find the resultant sum of two or more displacement vectors. Each displacement vector will be in the form `(x,y)` and separated by a single space.

Output

For each problem, you will print the resultant vector in the form `(x,y)` on a single line with no spaces.

Example Input File

```
3
(3,4) (-1,3)
(5,2) (-6,0) (6,-2)
(19,12) (4,-2) (5,6) (1,4)
```

Output to Screen

```
(2,7)
(5,0)
(29,20)
```

12. Widget Wackiness

Program Name: Widget.java **Input File: widget.dat**

<commercial> Walter Widgeton here announcing Widget, Inc.'s biggest deal! Bring in five widget wrappers and get one free widget! Don't settle for any old thingamajig! Come on down and buy our widgets for $2 each!</commercial>

Given an amount of money, the cost of a widget, and the number of widget wrappers needed to get a free widget, determine the number of total widgets you can acquire.

Input

- The first line will contain a single integer `n` that indicates the number of data sets to follow.
- Each data set will consist of a line containing three integers, `m`, `c`, `w`, with:
  - `m` = the amount of money in dollars you have, 0 <= `m` <= 100, `c` = the cost in dollars of a single widget, 1 <= `c` <= 100
  - `w`= the number of widget wrappers you can trade in to receive a free widget, 2 <=w <= 100

Note that all widgets, purchased or free, are in wrappers.

Output

For each data set in the input, output a single line "`X`", where `X` is the number of widgets you can acquire by continually trading in wrappers for widgets until you have redeemed as many wrappers as possible.

Example Input File

```
3
10 2 5
7 1 4
56 2 4
```

Example Output to Screen

```
6
9
37
```