A+ Computer Science
Computer Science Competition
Hands-On Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 12.

2. All problems have a value of 60 points.

3. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros. Unless otherwise specified, your program should read to the end of file.

4. Your program should not print extraneous output. Follow the form exactly as given in the problem.

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Point Values and Names of Problems

Number	Name
Problem 1	Twin Puppies
Problem 2	The Ultimate Formula
Problem 3	Maze Runner
Problem 4	Fibonacci
Problem 5	Mirror, Mirror
Problem 6	Scientific Notation
Problem 7	Talent Counter
Problem 8	Digits
Problem 9	Word Classification
Problem 10	Filtering
Problem 11	Royal Flush
Problem 12	Basketball League

1. Twin Puppies

Program Name: Puppies.java **Input File: None**

Your two dogs, Doggo and Puppo, are the most adorable dogs in the world, and a world-renowned puppy magazine wants you to draw a picture of your puppies for the front page of the magazine! However, you lack the finances to buy art supplies to draw a painting of your puppies, so you must use your computer to draw it instead. Output the picture of the two dogs below.

Input

None

Output to Screen

```
01234567890123456789012345678901234567890123456789012345
             /)-_-(\          /)-_-(\
              (o o)            (o o)
     .------__/\o/              \o/\__------.
    /   __        /                \         __   \
\__/\ /   \_\ |/                    \| /_/   \ /\__/
     \\      ||                       ||       \\
     //      ||                       ||       //
     |\      |\                       /|      /|
```

(The numbers in the above output are not to be printed; they are simply for reference.)

2. The Ultimate Formula

Program Name: Formula.java **Input File: formula.dat**

Your immense knowledge of mathematics allowed you to derive the ultimate formula to life – a five parameter formula that determines how fulfilling your life will be. It truly is a masterpiece of mathematics, but you don't have the knowledge to do your friends' fulfillment calculations by hand. Write a computer program that will take in the five necessary parameters and output a person's fulfillment value.

```
Fulfillment = |a*sin(b)+(c/d)-e|
```

Input

The first line will contain a single integer `n` that indicates the number of lines that follow. The next `n` lines will each contain a person's name followed by the five integers needed to calculate their fulfillment value.

Constraints:

- The fulfillment value `f` will be in the range `0 <= f <= 2^63 - 1`.
- `-2^31 <= a, b, c, d, e <= 2^31 - 1`
- `d` will not equal 0.
- `a, b, c, d, e` are all integers.

Output

Output the person's name, followed by a space, followed by their decimal fulfillment value, rounded to three decimal places.

Example Input File

```
2
Sammy 3 6 15 2 14
Ben 2 60 324 6 8
```

Example Output to Screen

```
Sammy 7.338
Ben 45.390
```

3. Maze Runner

Program Name: Maze.java **Input File: maze.dat**

You are a maze runner and you are stuck inside of the maze! You only have a certain amount of time to get out of the maze before the doors close, and if you don't get out you will be captured by the guards who patrol the maze at night! The maze is a grid of squares and each movement you make in the maze takes two minutes. Write a program that will determine if it's possible to make it out of the maze in time!

Input

The first line will contain a single integer n that indicates the number of data sets that follow. Each data set will contain the following information in this order:

- The first line will contain three integers, `i` and `j`, which indicate the size of the maze (where `i` is the number of rows and `j` is the number of columns), and `k`, which represents the number of minutes the maze runner has to escape the maze.
- The next `i` lines will contain `j` characters that represent the contents of the maze.

Maze Information:

- `W` – a wall, you cannot walk on these squares
- `.` – a path, you can walk on these squares
- `S` – start position, you begin the maze on this tile
- `E` – exit position, you end the maze on this tile

Constraints:

- `2 <= i,j <= 15`

Output

If you can exit the maze under the time constraints, then print the number of minutes you had left before the maze doors closed. If you can't escape the maze in time or the maze is unsolvable, then print `-1`.

Example Input File

```
2
4 4 15
S.WW
...W
WW..
E..W
2 2 2
S.
.E
```

Example Output to Screen

```
1
-1
```

4. Fibonacci

Program Name: Fibonacci.java **Input File: fibonacci.dat**

Your knowledge of the beloved Fibonacci sequence allows you to not only determine numbers of the sequence but add any subset of Fibonacci numbers together! Given a series of positions in the Fibonacci sequence you must add together, output the sum of all of the respective numbers.

The Fibonacci sequence consists of adding the two previous numbers together to get the next number in the sequence. The first numbers of the sequence as follows: 1, 1, 2, 3, 5, 8, …

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will contain an integer `m` that indicates the amount of numbers you will need to add together. On the next line, there will be `m` integers indicating what positions of the Fibonacci sequence you will need to add together (the first number in the sequence has position 1).

Note: The positions you must add are not necessarily in ascending order and are not guaranteed to be unique.

Output

For each data set, your output should be the sum of the numbers in the Fibonacci sequence.

Example Input File

```
2
3
1 3 5
6
7 2 4 1 3 7
```

Example Output to Screen

```
8
33
```

(For the first test case, we must add the 1st, 3rd, and 5th numbers in the Fibonacci sequence together. This is 1 + 2 + 5, which equals 8.)

5. Mirror, Mirror

Program Name: Mirror.java **Input File: mirror.dat**

There is a giant mirror in your room and any piece of writing you put in front of it gets reversed! However, this mirror appears to be a magic mirror on the wall that doesn't change the order of the words – the order of the words will stay the same but each individual word gets reversed!

Input

The first line will contain a single integer `n` that indicates the number of lines that follow.
Each of the following `n` lines will contain a sequence of words that are to be reversed but kept in the same order.

Output

For each of the `n` lines of input, print a line where every word on the line is reversed.

Example Input File

```
3
Hello!
Coding is so much fun
+A retupmoC ecneicS si eht tseb ynapmoc ni eht dlrow
```

Output to Screen

```
!olleH
gnidoC si os hcum nuf
A+ Computer Science is the best company in the world
```

6. Scientific Notation

Program Name: Science.java **Input File: science.dat**

Science has never been your strong suit, especially when you try and combine it with math. That's why you have no idea what scientific notation actually means! Instead of writing numbers in scientific notation, you'd just prefer to know what the actual value of the number is.

Scientific notation is a way of simplifying numbers to a more concrete form by turning a number into the form `a*b^c`. We are trying to do the opposite here by taking three numbers from the given form and converting it to the actual number.

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 1 double, `a`, and 2 integers, `b`, and `c`, each separated by a space.

Constraints:

- `-2^63 <= (a*b^c) <= 2^63-1`

Output

Output the actual value of the scientific notation for each data set, assuming the three numbers are placed in the form `a*b^c`, rounded to the nearest integer.

Example Input File

```
2
2.5 10 3
4.1 8 4
```

Example Output to Screen

```
2500
16794
```

7. Talent Counter

Program Name: Talent.java **Input File: talent.dat**

You're a teacher of a class of talented students, and each student has a certain talent, each one marked with a number. The talent show is coming up, and each class must try and pack the most talent into the smallest act to get the most points. Your job is to find the smallest consecutive subset of students that have the number of different desired talents for your act.

Time Limit

For each data set, your program must have a runtime of `O(Nk)`, where:

- `N` = the number of students in the class
- `k` = the number of different talents required in the subset

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 two lines of input. The first line will contain a single integer that indicates the number of different talents you need in your subset. The second line will contain an unspecified number of integers, indicating each of the students' talents.

All of the student's talents will be integers, and in the range `1 <= talent <= k`, where `k` is the number of desired talents in the class.

Output

Output the length of the smallest consecutive subset of students that have the desired number of different talents. If there is no subset of students that has the number of talents you require, output `-1`.

Example Input File

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

Example Output to Screen

```
3
5
```

8. Digits

Program Name: Digits.java **Input File: digits.dat**

You've taken a recent interest into finding how small you can make a number through summing its digits over and over again until you're left with a single digit. For example, the digits of 8966 would sum into 29, which sums into 11, which sums to 2.

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 a single integer, which will be greater than or equal to zero.

Output

Sum the digits of the given number over and over until you are left with a single integer, and then output that integer.

Example Input File

```
4
8966
273
548702
12345
```

Example Output to Screen

```
2
3
8
6
```

9. Word Classification

Program Name: Words.java **Input File: words.dat**

While math and science may be your strong suits, learning how to work with words is also a very valuable skill, since knowing how to read and write is almost a necessity these days. You'll be tasked with taking in a sequence of words and finding the longest word, the shortest word, and the most common letter in the sequence (not case sensitive).

Any punctuation (commas, semicolons, apostrophes, colons, periods, exclamation points, quotation marks, and question marks) does not count toward the length of a word, and each test case will have exactly one word that is the longest, one word that is the shortest, and one letter which appears the most.

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 a sequence of words on one line.

Output

For each test case, output the longest and shortest words and the word with the most duplicate letters in the following format, where `n` is the test case you are currently evaluating:

```
Case #n:
Longest: <longest_word>
Shortest: <shortest_word>
Letter: <common_letter>
```

For the most common letter, you should print a lowercase letter even if some (or all) of the occurrences are uppercase. However, the longest and shortest words should be printed as they appear in the input data (excluding punctuation).

Example Input File

```
2
I only love writing code in Mississippi.
Computer science is the best subject!
```

Example Output to Screen

```
Case #1:
Longest: Mississippi
Shortest: I
Letter: i
Case #2:
Longest: Computer
Shortest: is
Letter: e
```

10. Filtering

Program Name: Filter.java **Input File: filter.dat**

After doing the previous problem, you've gotten quite good at looking at words and finding differences between them. You've been hired to write a segment for the local newspaper as a result, and you want to sound as smart as possible in the article you write by only adding big words to the article. The bigger the word, the higher your IQ is, right?

Input

The first line will contain a single integer `n` that indicates the number of lines to follow.
Each data set will contain two lines. The first line will contain a single integer that indicates the minimum length of the words you are filtering. The second line will contain a sequence of words separated by spaces.

Output

Output all of the words in the sequence that are at least as long as the minimum length you require. Sort the words in alphabetical order and separate them with spaces (not case sensitive, but print the words in the same way they are given to you).

Example Input File

```
2
5
Encyclopedia brother also suit Samuel beachside
9
Houston requirement flag exponential Pleasantries
```

Example Output to Screen

```
beachside brother Encyclopedia Samuel
exponential Pleasantries requirement
```

11. Royal Flush

Program Name: Royal.java **Input File: royal.dat**

In poker, a royal flush is the best possible hand you can attain in the game. It is unbeatable – if you have a royal flush, you automatically win the hand! You get a royal flush when you have an Ace, King, Queen, Jack, and a Ten all in the same suit (either Clubs, Spades, Hearts, or Diamonds). Write a program that will determine if you have a royal flush by looking at the two cards in your hand and the five cards on the table (a royal flush can be attained by any five of these cards).

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will contain seven lines. Each line will contain a letter (indicating the value of the card) followed by a space, and then the suit of the card. Values of two through nine will be in numerical form (2-9) while the rest of the values will be labeled as follows:

- Ten: T
- Jack: J
- Queen: Q
- King: K
- Ace: A

Output

Output "`Royal Flush!`" if there is a royal flush on the table. Otherwise, output "`Poker is hard`".

Example Input File

```
2
A Clubs
4 Diamonds
K Clubs
T Clubs
6 Hearts
J Clubs
Q Clubs
A Diamonds
A Clubs
A Hearts
A Spades
K Diamonds
J Diamonds
Q Diamonds
```

Example Output to Screen

```
Royal Flush!
Poker is hard
```

12. Basketball League

Program Name: Basketball.java **Input File: basketball.dat**

You decided to quit your job and leave your math, science, and English knowledge behind you to become the commissioner of your local basketball league. The problem is, you have absolutely no idea how to rank the teams based on their record! Write a program that will take in the game results of the league and rank the teams accordingly.

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 several lines. The contents of each dataset are as follows:

- The first line will contain a sequence of one word team names, separated by a space.
- The second line will contain a single integer m indicating the total number of basketball games played in the league, and also the remaining number of lines in the data set.
- The following m lines will contain three team names – the first two indicating the two teams that played each other in the game and the last one indicating the winner.

It is guaranteed that every team will play the same number of games in the basketball league. All of the team names will start with a capital letter, and the rest of the letters will be lowercase.

Output

First, print "`League #n:`" where `n` is the current test case you are examining.
Output all of the basketball teams in the following format:
"`<team_name> <wins>:<losses>`"

The teams should be printed in order based on the number of wins a team has. If two teams have the same number of wins, sort them alphabetically and print them in that order.

Example Input File

```
1
Rockets Mavericks Thunder Warriors
6
Rockets Mavericks Rockets
Thunder Warriors Thunder
Rockets Thunder Rockets
Rockets Warriors Rockets
Mavericks Thunder Thunder
Mavericks Warriors Mavericks
```

Example Output to Screen

```
League #1:
Rockets 3:0
Thunder 2:1
Mavericks 1:2
Warriors 0:3
```