10. Kanye 2020

Program Name: kanye.java **Input File: kanye.dat**

Regardless of the outcome of the 2016 election, it is undeniable that in 2020 Kanye West will be elected president in a landslide vote and lead our country even further into greatness. Election day in 2020 falls on November 3rd, write a program to determine how many days from a given date until Kanye is elected president.

Input

The first line will contain a single integer n that indicates the number of data sets that follow.
the following n lines each contain a test case in the format MONTH DAY, YEAR.
keep in mind that February has 29 days every year that is divisible by 4 and not 100, as well as every year divisible by 400.

Output

if the given date is on or after election day 2020, print "`Kanye in the house`". Otherwise, print the number of days until election day 2020, in the format "`x day(s) until Kanye`".

Example Input File

```
3
November 3, 2020
November 2, 2019
August 2, 2018
```

Example Output to Screen

```
Kanye in the house
367 day(s) until Kanye
824 day(s) until Kanye
```

January	31
February	28/29
March	31
April	30
May	31
June	30
July	31
August	31
September	30
October	31
November	30
December	31

7. Neighbor

Program Name: Neighbor.java **Input File: neighbor.dat**

Two fractions are considered "neighbors" if the numerator of the reduced positive difference between the two fractions is 1. For example, the fractions 1/11 and 1/12 subtract to be 1/132, and are therefore "neighbors". However, 2/5 and 4/5 differ by 2/5, and are not neighbors. Determine and output either the "neighbor" fraction value, or output "NOT NEIGHBORS".

Input: The positive integer values for two fractions, all on one line, in the order numerator, denominator, numerator, denominator, separated by single spaces.

Output: The positive "neighbor" difference between the two fractions, or the phrase "NOT NEIGHBORS".

Sample input:

```
1 11 1 12
20 19 19 19
2 5 4 5
```

Sample output:

```
1/132
1/19
NOT NEIGHBORS
```

10. Say Cheese!

Program Name: Cheese.java **Input File: cheese.dat**

Satellites are abundant in earth's orbit, some designed to take high resolution photographs of our world. The launch day and time of these satellites is available, as well as the dates and times they would be over certain locations in the world, including your area. Given the launch time precise to the second, and a time the satellite would be directly over your location, you can calculate how long it will be from launch time to picture time, when you can look up into the sky and say, "Cheese!", to be a part of the satellite image!

For example, between a launch date of April 10, 2014 at 12:30:10 (military time), represented by the input string `041014123010`, and a "Say Cheese!" date of June 16, 2015, at 16:45:20, the total elapsed time is 432 days, 4 hours, 15, minutes and 10 seconds, represented by the output **`432:04:15:10.`**

Input: Several lines of data, with two values on each line, the first representing the "Say Cheese!" time for you, and the second the launch time of the satellite, each value in the form MMDDYYhhmmss, representing the Month, Day, Year, hour, minute, and second of the time.

Output: The span of time in days, hours, minutes, and seconds between the two given times. All times will be in the present millennium (year>2000). Assume also that all times are in standard time; no allowance for daylight savings times needs to be made. Output is to be in the format **dd:hh:mm:ss**.

Sample input:

```
061615164520 041014123010
061602164520 042002205030
053015185045 022014164530
```

Sample output:

```
432:04:15:10
56:19:54:50
```

A+ Computer Science – January 2016 Packet 1

1. DaysGoneBy

Program Name: Days.java Input File: days.dat

From January 1st to January 10th, 9 days have gone by. In a non-leap year, there are 364 days from January 1st to December 31st. From the first of the month of May to the end of that same month, 30 days have passed. From February 4th to March 4th in a leap year, 29 days have gone by.

Given a pair of three integer value sets, each representing a date in the same year, with the first set guaranteed to be earlier by at least one day than the second set, calculate and output the number of days gone by.

Input: Several pairs of three integer value sets (M, D, Y), representing the month, day, and year. for each date. Each set is on one line, and the Y value for each set will be any year 1700<=Y<=2020, but will be the same value for both dates.

Output: A sentence that reports the two dates and the number of days gone between the two dates.

Sample input:

```
1 1 2015
1 10 2015
1 1 1714
12 31 1714
2 4 2000
3 4 2000
```

Sample output:

```
There are 9 days gone by from 2015-01-01 to 2015-01-10.
There are 364 days gone by from 1714-01-01 to 1714-12-31.
There are 29 days gone by from 2000-02-04 to 2000-03-04.
```

A+ Computer Science – January 2016 Packet 12. Fruit Salad

Program Name: Fruits.java Input File: fruits.dat

Sally is throwing a party, and wants to make a fruit salad. Cost is of no concern to her, but the quality of the fruit at the store is, so she buys all of the fruits that she deems to be of good quality, whether or not she puts them in the salad. She also wants to try different combinations in the salad, perhaps not using all of the fruit she buys. She needs your help in showing her what possible fruit salads she can make.

Given a possible list of fruits, and the number of fruits to be used in the salad, show all the possible fruit combinations, each combination with the fruits listed in alpha order, and the entire list of combinations also alphabetized.

Input: Several data sets, each on one line, with an integer N representing the number of desired fruits in the salad, followed by a list of fruits. Each data set will be on one line, and each item in the data set will be separated by one space. There will be no duplicate fruits listed, and each fruit will be a single string, but the list of fruits may not necessarily be in alpha order.

Output: All of the possible combinations of fruit salad, with the fruits listed in alphabetical order for each combination, each separated by one space, and all combinations listed in alphabetical order. Each combination should be on one line, and each complete set of combinations should be followed by a blank line.

Assumptions: For each data set, 2<=N<= number of fruits purchased.

Example Input:

```
3 strawberry orange banana kiwi
10 pineapple orange tomato banana kiwi avocado strawberry blueberry mango cherry
2 banana kiwi strawberry blueberry cherry pineapple orange
```

Example Output:

```
banana kiwi orange
banana kiwi strawberry
banana orange strawberry
kiwi orange strawberry

avocado banana blueberry cherry kiwi mango orange pineapple strawberry tomato

banana blueberry
banana cherry
banana kiwi
banana orange
banana pineapple
banana strawberry
blueberry cherry
blueberry kiwi
blueberry orange
blueberry pineapple
blueberry strawberry
cherry kiwi
cherry orange
cherry pineapple
cherry strawberry
kiwi orange
kiwi pineapple
kiwi strawberry
orange pineapple
orange strawberry
pineapple strawberry
```

4. The Librarian

Program Name: Librarian.java **Input File: Librarian.dat**

You're a librarian at a local library. Each day, you receive a shipment of 10 books in a random order. Your job is to sort the books in order for them to be shelved correctly. Your library is unique in that the books are sorted alphabetically from the end of the title to the beginning. Write a program that takes in 10 books and returns them sorted in alphabetical order from the end of the title to the beginning.

Input

The first line of input will contain an integer `n`, which represents the number of test cases. Each test case will be 10 book titles, each a single word and each on a new line. There will be a blank line between each case.

Output

Print out the books, sorted in alphabetical order from the end of the title to the beginning.

Example Input File

```
1
MOBYDICK
ULYSSES
DONQUIXOTE
HAMLET
WARANDPEACE
THEODYSSEY
THEGREATGATSBY
THEADVENTURESOFHUCKLEBERRYFINN
THEILIAD
PRIDEANDPREJUDICE
```

Example Output to Screen

```
THEILIAD
WARANDPEACE
PRIDEANDPREJUDICE
DONQUIXOTE
MOBYDICK
THEADVENTURESOFHUCKLEBERRYFINN
ULYSSES
HAMLET
THEGREATGATSBY
THEODYSSEY
```

9. The Technician (WILSHIRE HINT: DYKSTRA!)

Program Name: Technician.java **Input File: Technician.dat**

Your job as a technician is to connect nodes together using cable. Your specialty is that you can use the least amount of cable to get from one node to another. Your task is to write a program that will return the smallest length of cable required to get from one node to another, given a number of nodes and their distances from other nodes.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will start with an integer `d` that indicates the number of distances between nodes. The next `d` lines will consist of two characters and an integer, all separated by a space, that denote two nodes and the distance between them. The final line for each data set will consist of two characters, the start node and the end node. There will always be a path from the start node to the end node.

Output

Output the smallest distance between the start node and the end node using only the distances specified in the input.

Example Input File

```
2
5
A B 3
B D 6
A C 7
A D 4
C B 2
A C
6
A B 1
B C 1
D C 1
E D 1
A D 4
A E 5
A E
```

Example Output to Screen

```
5
4
```

10. The Cashier

Program Name: Cashier.java **Input File: Cashier.dat**

As a cashier, you have to be very good with quick math and working with change. You decide that rather than having to do all of the math in your head, you can just write some code that can find the least amount of coins to use to return the change. Given the price of the item, the amount of cash paid, and the available denominations of coins, write a program that returns the least amount of coins, assuming that you have unlimited amounts of each denomination.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data set will be two lines. The first line will be two decimals, representing the price of the product and the amount of cash paid in dollars, respectively. The second line will be a list of the available coin denominations in cents, each separated by a space.

Output

Output an integer denoting the smallest amount of coins that you can use to create the correct amount of change, assuming you have an infinite amount of each coin denomination. Print `CANNOT CREATE EXACT CHANGE` if you cannot create the exact amount of change with the coins given.

Example Input File

```
3
15.00 20.00
1 5 10 25 50
6.75 9.50
1 4 8 16
0.01 0.02
2 3 4
```

Example Output to Screen

```
10
20
CANNOT CREATE EXACT CHANGE
```

A+ Computer Science – October Packet 1 2016

11. The Gamer

Program Name: Gamer.java **Input File: Gamer.dat**

You're playing your favorite puzzle game, but you've gotten to a level that you can't figure out. In this game, your character must traverse a 3d maze in a futuristic setting. In order to escape, you will have to phase through walls using a special device to get past them. To add a level of difficulty, only certain walls can be phased through and your device only has a certain number of uses. The object of the game is to find the quickest path to the exit. Moving a space takes one second (You can move in any of the four cardinal directions), climbing up or down a floor takes four seconds, and phasing into a wall takes two seconds in addition to the time it took to get to that space. You can phase into walls that are above or below you.

Input

The first line will contain a single integer `n` that indicates the number of data sets that follow. Each data starts with a line in the format `f r c n`, where `f` is the number of floors, `r` is the number of rows on each floor, `c` is the number of columns on each row, and `n` is the number of uses your device has. The next `f` floors will consist of `r` lines each with `c` characters on each line. The mazes will consist of the characters `.` `0` `#` `S` and `E`. The `.` denotes a clear space, the `0` denotes a wall that can be phased through, the `#` denotes a wall that cannot be phased through, the `S` denotes the starting space, and `E` is the exit. There will be no empty spaces between cases or between floors.

Output

Output the least amount of time it takes to get from the start to the end of the maze in seconds, `S` to `E`. If you can't reach the exit, output `IMPOSSIBLE LEVEL`.

Example Input File

```
2
1 4 4 3
S..#
.#.#
.#0#
000E
3 1 1 0
S
0
E
```

Example Output to Screen

```
10
IMPOSSIBLE LEVEL
```

UIL – Computer Science Programming Packet – District 2 – 2016 UIL

1. Aiguo

Program Name: Aiguo.java **Input File: aiguo.dat**

Aiguo is an avid reader and really loves grammar. He enjoys reading proofs of works, and gets to make changes to these works before they get published! However, he has noticed that it is incredibly time consuming to really highlight the changes he makes in these electronic versions. What Aiguo really wants is the ability to just make the changes he sees fit with his expertise and have a program analyze the changes he made to report to the publisher.

The changes that Aiguo makes can be represented by deletions or additions to the original text. He gets an initial work, then will either add or remove words and characters to improve and revise the works. He would like you to write a program that can identify which characters he removed and added by looking at the initial and final version of his work.

Input: The first integer will represent the number of data sets to follow. The first line of each data set will be the initial work, before Aiguo's changes. The second line of each data set will be the final work, after Aiguo's changes.

Output: Each data set should have 3 lines of output. The first, labeled "added: " will be the characters that Aiguo added to the initial text. The second, labeled "removed: " will be the characters that Aiguo removed from the initial text. The last, labeled "unchanged: " will be the characters that Aiguo did not change. Each of these should be listed in the order that the changes were made. If there exists an exchange where it was ambiguous if a correction was an additional of one word or the removal of another, choose to state the correction that comes first in the text.

Assumptions: Each work will be on one line.

Sample Input:

```
3
the brown fox curiously jumped over the sad dog
the quick brown fox jumped over the lazy dog
Ceci n'est pas ma femme.
Ceci est ma femme.
I love chocolate!
He loves chocolates!
```

Sample Output:

```
removed: curiously sd
added: quick lzy
unchanged: the brown fox jumped over the a dog
removed: n'pas
added:
unchanged: Ceci est ma femme.
removed: I
added: Hess
unchanged: love chocolate!
```

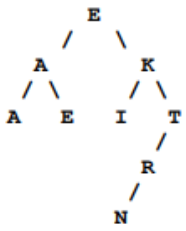
6. Felipe

Program Name: Felipe.java **Input File: felipe.dat**

Felipe is fascinated with binary search trees, especially with the four different traversals he just learned in class: inorder, preorder, postorder, and reverse order.

According to the rules he learned in class about building a character tree using a word, he practices doing this with his friend's names, and especially enjoys using his beautiful girlfriend's equally beautiful name, EKATERINA, with whom he is head-over-heels in love!

He starts with the first letter, E, as the root, and then sees that the next letter, K, is alphabetically after E, so he inserts it as a right node to the E, and then the A as a left node to the E. When he gets to the T, he goes right at the E, and then right again at the K, and proceeds on through the rest of the letters of her name, resulting in the binary search tree shown below. He decides to allow duplicate letters and slides those to the left as they reach a matching node.



He then proceeds to traverse in order, which means start at the top of the tree, going as deep to the left along each branch, and only outputting a node when reaching the end of a branch (a leaf node) or outputting a node after its left branch has been completely processed.

The inorder traversal (alpha order) of this tree would be: **AAEEIKNRT**
The preorder traversal (output a node BEFORE traversing either the left or right subtrees) would be: **EAAEKITRN**
Postorder traversal (output after both branches are completely traversed): **AEAINRTKE**
The reverse order traversal is just the inorder traversal backwards, which Felipe thinks about for a bit, and then realizes what to do, resulting in: **TRNKIEEAA**

Input: Several uppercased names containing no other symbols , each on one line, and each followed by a single character indicating the traversal to be performed: **E**(preorder), **O** (postorder), **I** (inorder), **R** (reverse order).

Output: The indicated traversal for each name, as shown in the examples.

Sample Input:

```
EKATERINA I
EKATERINA E
EKATERINA O
EKATERINA R
```

Sample Output:

```
AAEEIKNRT
EAAEKITRN
AEAINRTKE
TRNKIEEAA
```

7. Gary

Program Name: Gary.java **Input File: gary.dat**

Gary has just encountered boolean postfix expressions in class, and is a bit confused. He understands the easy ones, like, "true and false" becomes "true false and", and knows how to evaluate them according the following rules of thumb:

- "and": evaluates to "false" only when both operands are "true"
- "or": evaluates "false" only when both operands are "false"
- "xor": evaluates to "true" only when the operand values are opposite
- "not": reverses the value from "true" to "false", or vice versa

It's evaluating the more complex ones in postfix form that give him a bit of trouble.

For example, "true or false and true" is confusing to him. *"What is the postfix version?"*, he wonders, *"and then what is the answer?"* Well, his teacher helps him out and just gives him the postfix version, for now, which is "true false true and or", and then shows him how to process it.

His teacher says, *"Take the first two boolean values and see if there is an operator immediately after them. If there is, do that operation, and replace the result in place of that expression. If not, ignore the first operand for now, and look at the next pair to see if there is an operator following, and evaluate it and replace it with an answer."*

"In the example, "true false true and or", there is no operator for the first two operands, so you ignore the first operand and look at the next two, which have an "and" after them. Well, "false and true" evaluates to "false", so you replace the "false true and" with just "false", which now gives you the expression, "true false or", which is "true", and is the final value of the expression.

"If you have a "not" in the expression, like this one, "true false or not", the result here would first be, "true not", since "true false or" becomes "true", and then "false", since the "not" operator only needs one operand, and "not true" means "false"."

Input: Several lines of data, each line with a postfix boolean expression made up of the words, "true", "false", "not", "and", "or", and "xor", with no parentheses, all separated by single spaces.

Output: The final **true** or **false** value of the postfix boolean expression.

Sample input:

```
true true or
true false and
true false true and or
true false or not
```

Sample output:

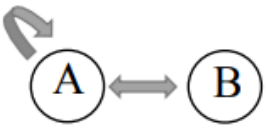
```
true
false
true
false
```

2. Nicholas

Program Name: Nicholas.java **Input File: nicholas.dat**

Nicholas is learning about how to manually count path lengths in a graph, such as the one shown here, using adjacency matrices, but still needs your help to confirm his counting. He started with this simple example, which produces the following adjacency matrix:

```
  A B
A 1 1
B 1 0
```



He has learned that this matrix means that given two vertices A and B in the graph above, there is a connection from A back to itself, and a two-way connection from A to B. To count the number of paths of length one, or direct connections in the graph, all he has to do is count the number of 1s in the graph, three in this case, represented in letter notation as AA, AB, and BA. AA means that the connection starts and ends at A, AB means it starts at A and ends at B, and so on.

However, counting the number of two-hop paths is a little more involved. He can see that AAA is a possibility, and then ABA, and then BAB, but wonders if AA could be combined with B. His teacher says, "Yes, it can.", so he adds AAB and BAA to his count, making a total of five 2-hop paths.

"What about 3-hop paths?", he wonders. Hmmm, let's see, starting from A there would be AAAA, AAAB, AABA, ABAA, and ABAB. Starting from B, the 3-hop paths are BAAA, BAAB, and BABA. Altogether, that would make eight 3-hop paths within this graph.

Your job is to take any graph with at least two, but no more than five vertices, and help Nicholas confirm his counting of the number of paths of any length from 1 to 3 hops.

Input: Several lines of data, each consisting of an integer N, indicating a graph with N nodes, followed by an integer P, the path length to calculate, followed by N strings, each of length N, containing 0s and 1s, indicating the adjacency matrix for the graph.

Output: The number of P length paths in the given matrix.

Sample Input:

```
2 1 11 10
2 2 11 10
2 3 11 10
3 1 111 001 001
4 2 0100 0010 0001 1000
```

Sample Output:

```
3
5
8
5
4
```

3. Collation

Program Name: Collation.java **Input File: collation.dat**

In preparation for teaching her new kindergarten class, Agnes has made some devious plans. Rather than teaching the children the alphabet in the standard alphabetical order, she's decided to teach the kylographical order. For each test case you will print the words sorted lexicographically using kylographical order.

Input
The first line of input contains T, the number of test cases.

Each test case contains three lines. The first line in each test case is the kylographical order of the alphabet. The next line contains a single integer, *N*, the number of words to be sorted. The last line contains *N* space-delimited lowercase words.

Output
For each test case, print the words sorted lexicographically using kylographical order.

Constraints
```
1 <= T <= 10
1 <= N <= 20
```

Example Input File
```
3
ojhdfnexkizuyvgltraqpwbsmc
5
dance safe sidewalk orange safety
zyxwvutsrqponmlkjihgfedcba
7
cage hammer tree cow yearning treatment morning
fhgdsakjlpioytuerqwczxbnvm
3
apple justice favorite
```

Explanation of Output
The letter *o* appears first in the first given kylographical order, so *orange* comes before all the other words. Since the letter *d* appears before the letter *s*, *dance* is the next word. *sidewalk*, *safe*, and *safety*, all begin with an *s*, so they are compared with the next letter. Since *i* appears kylographically before *a*, *sidewalk* is next. Lastly, since *safe* is a sub-word of *safety*, and therefore shorter in length but equal up to its length, *safe* comes before *safety*. In the second kylographical order (the standard alphabet backwards) the output places the words in reverse standard alphabetical order.

10. Permutations

Program Name: Permutations.java **Input File: permutations.dat**

A permutation of a word is one in which the letters have been shuffled but the set of letters do not change. Given a word, find out how many distinct permutations of the word there are. If two permutations are lexicographically equivalent, they are not distinct.

Input
The first line of input contains T, the number of test cases.

The next T lines each represent a test case. Each test case consists of a single alphabetical lowercase word.

Output
For each test case, print the number of distinct permutations of the word there are.

Constraints
```
1 <= T <= 50
1 <= number of permutations <= 10^9
```

Example Input File
```
3
tall
splat
bug
```

Example Output to Screen
```
12
120
6
```

Explanation of Output
The word *tall* has the following 12 distinct permutations: *tall, tlal, atll, altl, ltal, latl, lalt, ltla, llat, llta, allt, tlla*.

UIL 215 Inv B
(Can be done dynamically)

2. Bitmap

Program Name: Bitmap.java **Input File: bitmap.dat**

A bitmap is a rectangular grid of bits. A bit may have one of two values, most commonly represented as `0` and `1`. In this problem, you will be given a bitmap of `0`s and `1`s, and you will have to determine the area of the largest rectangle that is filled with `1`s.

Input

The first line of input consists of two space-separated integers, *m* and *n*, which represent the dimensions of the bitmap, where *m* is the number of rows and *n* is the number of columns. This will be followed by *m* lines of *n* bits each.

Output

The output will be a single line giving the area of the largest rectangle that is filled with only 1s. You may think of the area as the total number of 1s in that rectangle.

Constraints

```
1 <= m <= 1000
1 <= n <= 1000
```

Example Input File

```
5 24
110010010000011111111011001000110
001010001001101111110100011010111
100111101011100001000011001011100
101101011101010101111110100000110
001101101101110000100011001101101
```

Example Output to Screen

```
10
```

---

4. Consonant Runs

Program Name: Consonant.java **Input File: consonant.dat**

Given a string, find the longest run of consonants, case insensitive and which may or may not be consecutive, that are either ascending or descending. In this problem the letter "Y" is a consonant. The following consonants are ascending – "b g t". The following consonants are descending – "m h c". Repeated consonants can be considered ascending when determining ascending runs and descending when considering descending runs. A string of a single consonant can be considered either ascending or descending.

For example, in the string

```
A human must turn information into intelligence or knowledge.
```

the consonants are

```
h m n m s t t r n n f r m t n n t n t l l g n c r k n w l d g
```

Excluding shorter substrings, the ascending runs are

```
hmn, mstt, r, nn, fr, mt, nnt, nt, ll, gn, cr, knw, dg
```

and the descending runs are

```
nm, ttrnnf, rm, tnn, tn, tllg, nc, rk, wld, g
```

The string `ttrnnf` is the longest ascending or descending string, thus the longest run is 6.

Input

The first line of input will consist of a single integer, *n*, indicating the number of lines to follow. Each line will consist of a string between 1 and 128 characters long, inclusive. The string may be a mix of uppercase and lowercase letters, digits, punctuation marks, and spaces.

Output

For each input string, print a single integer on its own line representing the count of the longest run of ascending or descending consonants in the string.

Constraints

```
1 <= n <= 100
```

Example Input File

```
3
A human must turn information into intelligence or knowledge.
$234.56
QWERTY
```

Example Output to Screen

```
6
0
3
```

1. Akio

Program Name: Akio.java **Input File: akio.dat**

Akio loses things frequently. At the end of the night of partying out on the town, Akio realizes that he is missing a lot of his things. Luckily, people have called him to let him know where all of his things are. But, since it is late and all of the places are closing, he needs the shortest path from the door of each place to his lost item in the room so he can get in and out quickly.

All the rooms are represented with walls/tables as **"#"** indicating places where Akio cannot walk. All places where Akio can walk are indicated with a **"."** The item in the room that Akio is seeking is indicated with an **"o"**. Akio always enters the room where the **"."** is on the edge of the room, indicating the door. In each room he has only left one item, but some of the rooms can have multiple doors, so you must always choose the closest door for the path for Akio.

For example, in the room shown below, the shortest path to get to the item is from the first door at the top of the diagram. The solution on the right shows this path with the bolded numbers marking the steps.

```
####.##.##
###.....##
###...####
###.######
#o..######
###.######
###......#
###.######
###.....##
#######.##
```

```
####1##.##
###32...##
###4..####
###5######
#876######
###.######
###......#
###.######
###.....##
#######.##
```

To help Akio, you don't need to show him the steps, but just tell him the length of the shortest path.

Input: The first integer will be the number of data sets to follow. Each data set is made up of a 10x10 grid of characters representing one of the rooms where Akio has left one item, as described above. Each 10x10 grid will be followed by a **"-"** for separation.

Output: For each room map, output the length of the shortest path from a door to his lost item.

Sample Input:

```
3
####.##.##
###.....##
###...####
###.######
#o..######
###.######
###......#
###.######
###.....##
#######.##
-
##########
###....o##
###...####
###.######
#...#....#
###.######
###.#....#
###.######
###.....##
#######.##
-
#######.##
#.......##
#.########
#.........#
########.#
#.........#
#.########
#.########
#......o##
##########
-
```

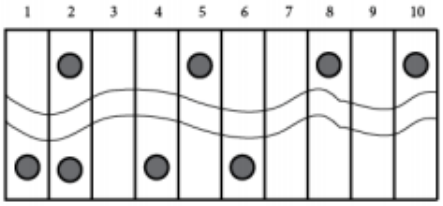
Sample Output:

```
8
17
35
```

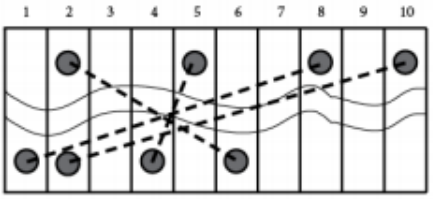
3. Dong-Sun

Program Name: DongSun.java **Input File: dongsun.dat**

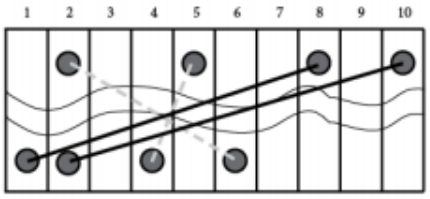
Dong-Sun is currently employed as a civil engineer building bridges in an area with an east-west running river and many cities on either side. These cities all have a lot of traffic between them, and many have filed bridge applications with his office. However, when looking through the applications, he's noticed that some have been filed that would intersect, which cannot be done, for obvious reasons.
The cities in Dong-Sun's area are labelled by their position along the north or south bank of the east-west running river. For example, here is cities 2, 5, 8, 10 on the north bank; and 1, 2, 4, 6 on the south bank:



The bridge applications that Dong-Sun has are bridges from city 10 to 2, 5 to 4, 2 to 6 and 8 to 1, These proposed bridges are indicated by the following notation, `10 -> 2, 5 -> 4, 2 -> 6, 8 -> 1`, with the first city of each notation pair always the one on the north side of the river, and the second one always on the south side. It is guaranteed that any proposed bridge will cross the river. The two north/south cities at position two tried to submit a `2 -> 2` bridge proposal, but missed the deadline.



Dong-Sun cannot build all these bridges because he can't approve bridges that intersect, but he does want to approve as many as he can. In this scenario, the most number of bridge applications he can approve is two, the `10 -> 2` bridge and the `8 -> 1` bridge, since these are the only ones that can be built without any bridges intersecting.



Given the bridge applications that Dong-Sun receives, in the notation from ***`north side city -> south side city`***, determine which bridges he should approve in order to provide the maximum number of bridges built for use by the cities.

Input: The first line, N, will be the number of data sets to follow. The first line of each data set will be the number of applications, I, which Dong-Sun has received. The next I lines will be the bridge applications in the form ***`north side city -> south side city`***, where ***`north side city`*** and ***`south side city`*** will be integers representing their location along the river bank.

Output: For each data set, output the applications that Dong-Sun can approve, in the order they were initially given. The output should be prefaced by "Approved Applications: " followed by a comma separated list of the approved bridge applications.

Dong-Sun - Sample input:

```
3
4
10 -> 2
5 -> 4
2 -> 6
8 -> 1
10
3 -> 4
1 -> 2
2 -> 3
6 -> 7
10 -> 1
8 -> 9
4 -> 5
9 -> 10
5 -> 6
7 -> 8
6
1 -> 15
2 -> 8
8 -> 3
9 -> 2
10 -> 2
11 -> 15
```

Sample output:

```
Approved Applications: 10 -> 2, 8 -> 1
Approved Applications: 3 -> 4, 1 -> 2, 2 -> 3, 6 -> 7, 8 -> 9, 4
-> 5, 9 -> 10, 5 -> 6, 7 -> 8
Approved Applications: 9 -> 2, 10 -> 2, 11 -> 15
```

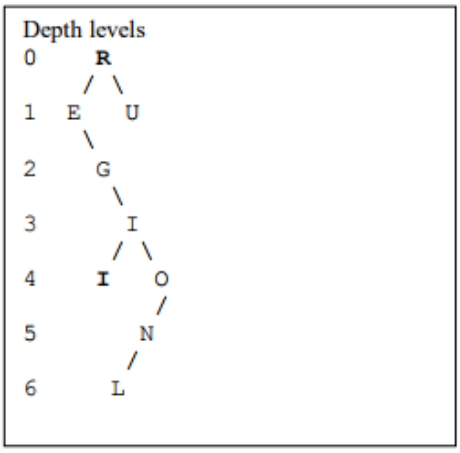
6. Fabiana

Program Name: Fabiana.java **Test Input File: fabiana.dat**

Fabiana has just learned about the depth concept as it relates to binary search trees and decides to do some research. She takes a single word, picks a letter from somewhere in the word, and finds the depth of where that letter settles into a binary search tree. Doing it by hand is not difficult, but is time consuming, and she needs your help in writing a program to do it for her to save her some time doing the research.

Given a single word with no spaces, followed by an integer P after a blank space, output the depth at which the letter in position P is inserted into a binary search tree.

Below is an example of a binary search tree formed by using the letters in the word REGIONUIL. The first letter is always the root of the tree, and then each subsequent letter "finds" its place relative to the root, and any other nodes in the tree, going to the left if its value is less than or equal to the current node, and going right if it is greater in value.



The root node that contains the letter R has a depth of zero, with the letters E and U at level 1, G at level 2, and so on.

The data set consisting of the word REGIONUIL and the value 0 means to find where the letter at position 0, the R, settles in the tree, which would be the root, or level 0.

The data set REGIONUIL 7 indicates the second occurrence of the letter I, which is inserted into the tree at level 4 as shown.

Input: A series of data sets, each on a separate line, consisting of a word (all uppercase) and an integer.
Note: The judges test data input will have different words than the one given here.

Output: The depth (or level) at which the indicated letter in the word settles in the binary search tree.

Sample Input:

```
REGIONUIL 0
REGIONUIL 1
REGIONUIL 3
REGIONUIL 5
REGIONUIL 7
```

Sample Output:

```
0
1
3
5
4
```

9. Manuela

Program Name: Manuela.java **Test Input File: manuela.dat**

Manuela is a space traveler. She regularly flies around the 'verse in her ship, a Jupiter-wing class 45X-328A. However, in all her travels, she is getting increasingly frustrated by all the various currencies she keeps having to carry around. As such, she would like your help to determine what is the best way to make change in all the different places she goes.

Given the names and the values of the currency of the place where she is arriving, and the amount she needs to make change for, tell her how much of each coin she should need to use the *least* number of coins.

Input: The first line, N, will be the number of data sets to follow. Each data set will begin with a number, i, representing the number of unique types of coins. The next i lines will be the name of each coin, followed by its value. After the ith line, there will be a final value representing the amount of change that needs to be made.

Output: Display the coins used, in alphabetical order, followed by a colon, and the number of each coin needed to use the minimum number of coins.

Sample Input:

```
3
6
two-dollar 200
dollar 100
quarter 25
dime 10
nickel 5
penny 1
337
7
aijika 13
uilika 11
quilika 7
yuilkia 5
dilkia 3
gilikia 2
tikki 1
14159
6
euro 100
pentuplepenny 5
quadruplepenny 4
triplepenny 3
doublepenny 2
penny 1
1027
```

Sample Output:

```
dime: 1 dollar: 1 penny: 2 quarter: 1 two-dollar: 1
aijika: 1089 gilikia: 1
doublepenny: 1 euro: 10 pentuplepenny: 5
```