1. AlphaFun

Program Name: AlphaFun.java Input File: alphafun.dat

Lester has a list of words that he wants to print, not in the usual alphabetical order but not in random order either. He has decided on a method to sort the words that he calls AlphaFun order.

The AlphaFun method sorts the words using the following procedure:

- first compare the 2nd letter of the words,
- then compare the 4th letter of the words (the 4th letter will be considered to be a space in words containing less than 4 letters),
- then compare the last letter of the words (the last letter will always be the last letter of the word, not a space),
- finally compare the first letter in the words,
- if all of the above are the same characters, then the words used for those letters are sorted alphabetically.

Note these examples:

Word	AlphaFun Characters
EGG	G GE
EGGS	GSSE
BREAD	RADB

Input

The input file contains an unknown number of lines, where each line of input contains a single word consisting of 3 to 10 letters.

Output

You will print the words in AlphaFun order.

Example Input File

```
BREAD
AERIE
BROAD
EGGS
EGG
WALLET
```

Example Output to Screen

```
WALLET
AERIE
EGG
EGGS
BREAD
BROAD
```

2. Boggle

Program Name: Boggle.java Input File: boggle.dat

Boggle is a word game played with 16 six-sided cubes that have a letter on each side of each cube. The 16 cubes are randomly rolled into a square 4x4 grid creating a puzzle. Each cube in the puzzle has exactly one letter facing up.

The object of the game is to find as many words as possible during a given amount of time. Words are formed using the following rules:

- Letters in a word must be adjacent horizontally, vertically or diagonally within the borders of the 4x4 grid.
- The letter on an individual cube can be used at most once per word.
- The same word can be used only once even if it appears more than once in the puzzle.

Points are scored by the length of the word as follows:

No. of letters:	**2 or less**	**3**	**4**	**5**	**6**	**7**	**8 or more**
Points:	0	1	1	2	3	5	11

Input

The first line of input will contain a single integer `n` that indicates the number of games to follow. For each game,

- The first four lines will each contain 4 uppercase letters of the alphabet that make up the puzzle.
- The next line will contain a single integer `m` that indicates the number of words that a player found in that puzzle.
- Each of the next `m` lines will contain a single word comprised of at most 15 uppercase letters.

Output

For each game, you will determine which words actually appear in the puzzle and print the total number of points scored in that game in the format: `PUZZLE #`*x*`:` *y* where *x* is the number of the puzzle and *y* is the number of points scored.

Example Input File (The Boggle cubes are in bold for readability purposes only.)

```
2
ERTD
HILS
DESK
TRAI
5
DESK
LID
RAISE
SCARED
LESS
REJO
ECIE
NERN
RATE
5
RACE
JOIN
REJOICE
RECREATE
TRAIT
```

Example Output to Screen

```
PUZZLE #1: 5
PUZZLE #2: 17
```

3. Bus Traffic

Program Name: Bus.java Input File: bus.dat

The DART bus service supervisor is doing research about the busses in his territory. He is following different busses and counting the number of passengers getting on and off the bus at each bus stop. He wants to know which part of the route has the fewest number of passengers on the bus.

To help with his data collection, he has given each section of a trip for a given bus a letter beginning with letter A and continues to use consecutive letters of the alphabet for the following sections of trip. At the first stop, he records the number of passengers who board the empty bus. That will be the number of passengers for section A. Then, at the next stop, he counts the number of passengers who get off the bus and the number who get on the bus to determine how many riders are on the bus for section B. He continues this method until the last stop that the bus makes in his territory. The last pair of numbers is considered still in his territory. When he completes his route, he reviews his numbers for each section to determine which section, or sections, has the fewest passengers.

Input

The first line of input will contain a single integer `n` that indicates the number of busses he will follow. For each bus, there will be a single line of data composed of integers separated by a space. The first integer will be the number of passengers getting on the empty bus at the initial stop. The remaining pairs of numbers contain the number of passengers getting off the bus (either 0 or a negative number preceded by a negative (-) sign) followed by the number of passengers getting on the bus. The negative number (or 0) will always be the first number in the pair since passengers must get off the bus before new passengers get on.

Output

For each bus followed, you will print, on a single line and separated by a space, an alphabetical list of the letter(s) of the section(s) with the fewest number of passengers and the number of passengers in the section(s).

Example Input File

```
2
12 -6 5 -5 8 -4 9 -7 7 -5 16 -2 8 0 5 -8 0 -11 7
23 -6 8 -8 5 -12 0 -5 9 -8 4 -12 14 -2 0
```

Example Output to Screen

```
B 11
D F H 10
```

4. Casting Out Nines

Program Name: Casting.java Input File: casting.dat

Before the days of hand-held calculators, many people used a method of 'casting out nines" to check their arithmetic when adding long columns of numbers. One method of casting out nines is to repeatedly (or recursively) add the digits in an integer until the sum of the digits is less than nine. The nines were cast out of each addend and added together and then cast out of that sum. Then the nines were cast out of the sum of the integers. The two "casts" were compared and if they were equal, the chances were pretty good that the sum was correct.

You are to write a program that will perform the task of casting out nines on an integer and report the "degree" of the cast. The degree of the cast is the number of times you have to add the digits until the sum is nine or less.

For example, consider the integer 59834467. The sum of its digits, 5+9+8+3+4+4+6+7, is 46, the sum of the digits in 46 is 10 and the sum of the digits in 10 is 1. Therefore, the degree of the cast would be 3.

Input

The first line of input will contain a single integer `n` that indicates the number of integers in the data file. Each of the following `n` lines will contain a single, positive integer no longer than 75 digits.

Output

For each integer input, you will print the degree of the cast for that number.

Example Input File

```
4
999999999
5
12345678991
999
```

Example Output to Screen

```
2
0
3
2
```

5. Decreasing Digits

Program Name: Decreasing.java Input File: decreasing.dat

Given a positive integer less than 100, there is a multiple of that integer that can be expressed using digits that are in non-increasing order. For example, 530 is a multiple of 53 and the digits 5, 3, and 0 are in non-increasing order.

You are to write a program that will find the smallest multiple of a given integer that can be expressed with digits in non-increasing order.

Input

The first line of input will contain a single integer `n` that indicates the number of test cases to follow. Each of the following `n` lines will contain a single integer `x` ($3 < x < 100$).

Output

For each number `x` input, you will print the smallest multiple of `x` with all digits in non-increasing order.

Example Input File

```
4
4
53
72
30
```

Example Output to Screen

```
8
530
432
60
```

6. Degree of Sorts

Program Name: Degree.java Input File: degree.dat

Some system analysts believe that one method to determine which sort might be the most efficient is to determine the ratio of the "degree" of the array to be sorted to the number of elements in the array. You are to write a program to determine the "degree" of an array of integers.

The "degree" of an array is found by:

- finding a sub-degree for each element of the array by determining the number of elements in the array with index values higher than that element that contain a value smaller than that element.

Array Index Value	0	1	2	3	4
Element Value	3	2	5	1	6
Element Sub-degree	2	1	1	0	-

- adding the sub-degrees of all the array elements together to get the "degree" of the array

Input

The first line of input will contain a single integer `n` that indicates the number of arrays to follow. Each of the following `n` lines will contain a list of integers that represent the values in the array that are to be sorted, each separated by a single space.

Output

You will output the degree of each array on a line by itself.

Example Input File

```
4
3 2 5 1 6
1 4 6 2 8 3 0 -1 2
4 3 2 1
6 5 4 1 2 3
```

Example Output to Screen

```
4
22
6
12
```

7. Fastest Mile

Program Name: FastMile.java Input File: fastmile.dat

James is keeping the times in the mile race for the swim team at his school. Before selecting the swimmers for the district swim meet, the coach wants to know each swimmer's fastest time from the mile swims that she has swum this year.

Input

The first line of input will contain a single integer `n` that indicates the number of swimmers to be considered. Each of the following `n` lines will contain the swimmer's first name with no spaces followed by a list of times in the form `mm:ss.hh` where $15 \leq$ `mm` ≤ 40 is the number of minutes, $00 \leq$ `ss` ≤ 59 is the number of seconds, and $00 \leq$ `hh` ≤ 99 is the number of hundredths of seconds. Each of the times will be separated by a single space.

Output

For each student, you will output the student's name and a space followed by her fastest time for the year in the format `mm:ss.hh` correct to the nearest hundredth of a second.

Example Input File

```
4
MARY 21:12.32 22:14.45 23:55.84 24:48.99
ANN 28:16.39 28:10.87 27:04.00 27:05.01 29:29.71
LUCY 33:43.82 28:24.95 30:52.24 29:55.76
SALLY 24:12.42 26:14.85 28:59.04 27:41.67 28:52.99 27:25.87
```

Example Output to Screen

```
MARY 21:12.32
ANN 27:04.00
LUCY 28:24.95
SALLY 24:12.42
```

8. Landscape

Program Name: Landscape.java Input File: landscape.dat

Roger is a software developer. Sometimes, he has to rotate pictures for his applications manually from landscape layout to portrait layout, one pixel at a time, so he can print the picture on a given printer. You are to write a program that will do this rotation for him.

Input

The first line of input will contain a single integer `n` that indicates the number of pictures he needs to rotate. For each picture, there will be 10 rows of pixels with 15 columns of pixels in each row representing a picture that is 3 inches wide and 2 inches long. There will be a blank line after each picture. Each pixel will be represented by a keyboard character.

Output

For each picture input, you will print the picture after it has been rotated counter-clockwise 90 degrees. Print a blank line after each picture printed.

Example Input File

```
1
222222222222222
$$$$$$$$$$$$$$$
%%%%____^^^%%%%
666666666666666
((((>>>.<<<))))
555555555555555
AAAAAAAAAAAAAAA
###############
000000000000000
111111111111111
```

Example Output to Screen

```
2$%6)5A#01
2$%6)5A#01
2$%6)5A#01
2$%6)5A#01
2$^6<5A#01
2$^6<5A#01
2$^6<5A#01
2$_6.5A#01
2$_6>5A#01
2$_6>5A#01
2$_6>5A#01
2$%6(5A#01
2$%6(5A#01
2$%6(5A#01
2$%6(5A#01
```

9. Multiplication

Program Name: Multiplication.java Input File: multiplication.dat

Ms. Appleworth teaches third grade math at Ward Elementary School. Her next lesson is on multiplication of large numbers and she needs you to write a program that will show the finished multiplication problem so she can check her students' solutions more quickly.

For example, if she wants the students to multiply 256 x 123, their solution would look like this:

```
   256
   123
   768
  512
256
31488
```

You are to write a program that will print the subproducts and the final product for her. The subproducts in this problem, in the order created when performing a standard multiplication, are `768`, `512`, and `256`. The final product is `31488`.

Input

The first line of input will contain a single integer `n` that indicates the number of problems. Each of the following `n` lines will contain two positive integers `r` and `s` ($10 < s < r < 100{,}000$) separated by a space.

Output

For each problem, you will print the subproducts in standard multiplication order followed by the product on a single line and separated by a space.

Example Input File

```
4
256 123
89 23
15468 3564
24 15
```

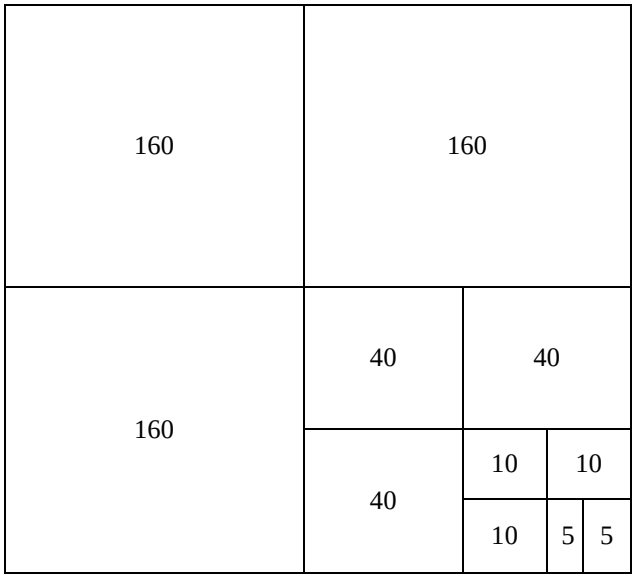
Example Output to Screen

```
768 512 256 31488
267 178 2047
61872 92808 77340 46404 55127952
120 24 360
```

10. Prospectors

Program Name: Prospectors.java **Input File: prospectors.dat**

Rob is a surveyor who is plotting the sections of land in an unmapped area of Alaska. A section of land covers one square mile and contains 640 acres of land. Rob has divided the section he is surveying into 128 horizontally or vertically contiguous, congruent, rectangular plots of land each containing 5 acres. Below is a diagram of a 640 acre section of land that has been subdivided as noted in the diagram and chart below.



Name	Acres	Shape
Section	640	Square
Quarter section	160	Square
Quarter quarter section	40	Square
¼ quarter quarter section	10	Square
Plot of land	5	Rectangular

To visualize his survey on paper, Rob has created a section grid that is 8 rows by 16 columns. Each cell of the grid represents one 5 acre plot of land. Some prospectors from the big Alaska gold rush have already staked claims on some of the 5 acre plots of land. Rob decided to denote those plots with capital letters of the alphabet using a unique letter of the alphabet for each prospector. You are to write a program that will tell how many acres are in each of the lots that lay unclaimed. A lot is a piece of land that contains all of the unclaimed plots that are completely surrounded by either the border of the section or by a prospector's claim.

Input

The first line of input will contain a single integer `n` that indicates the number of surveys to follow. Each survey will contain one section grid of 8 rows with 16 characters in each row. The characters in each row will be a capital letter (`A-Z`) indicating who owns the plot of land, or an asterisk (`*`) indicating an unclaimed plot of land. There will be a blank line following each section grid except for the last section grid.

Output

For each survey, you will output on a single line the number of acres available in each unclaimed lot from largest to smallest and separated by a space.

Example Input File (cont. on next page.)

10. Prospectors (cont.)

Example Input File

```
2
**AAAA****AA***B
*****AAAAAAA***B
CCCCCCDDD*******
********HHHHHHHH
*****FFFFF****H*
LL***FFFFF******
***WW********R**
***WW******ZZZZZ

****************
*******YXXXXXX**
******BB********
HHHHHHHHHHHHHHHH
**HHH*****D*****
****DDDD**DD****
****DD****DDD***
****DD****DDDD**
```

Example Output to Screen

```
135 110 65 35 20
195 75 70 70
```

11. Tree Compression

Program Name: TreeCompression.java **Input File: treecompression.dat**

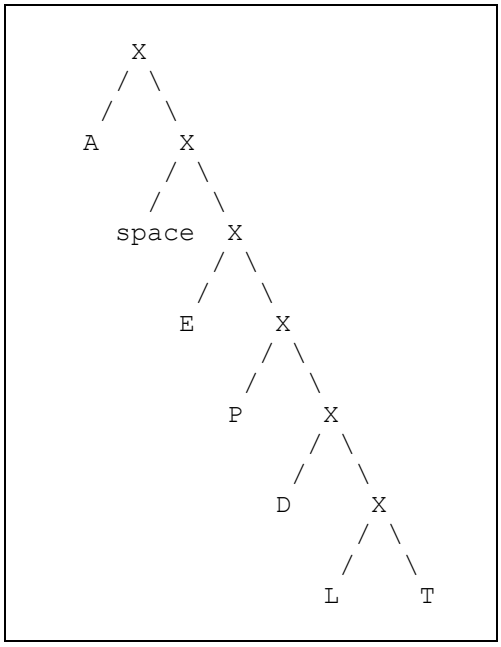
There are various methods for data compression that trade-off speed of compression and/or decompression with compression size. A relatively simple form of text compression, which we will call Tree Compression, involves using small bit sequences to represent the unique characters in a string and then creating a unique sequence representing each string.

The steps for this tree compression are:

- Count all of the unique characters in the string.
- Sort the characters by the number of times they appear in the sequence from most occurring to least occurring. Characters that occur the same number of times should be sorted by their ASCII numeric value.
- To insure that no two characters begin with the same bit sequence and to use as few bits as possible, assign a binary sequence to each character in your sorted list using the following method:
  - The last character in your list will always be all 1's.
  - If there is only a single character is your list, that character is assigned a 1.
  - Otherwise, if there is more than one character in the list, the first character in the list is assigned the binary value 0.
  - The second character, unless it is the last one, would be assigned 10.
  - The third character, unless it is the last one, would be assigned 110.
  - Continue this process until finally, as stated above, the last character has a 1 at the end instead of 0, i.e., it would be all 1's.

For example, for the string `ADA ATE APPLE` containing 4 `A`'s, 1 `D`, 1 `T`, 2 `E`'s, 2 `P`'s, 1 `L`, and 2 spaces, the encoding for each character is shown in the chart to the right.

Char	encoding
A	0
Space	10
E	110
P	1110
D	11110
L	111110
T	111111



By reading bits in, one by one, we can tell what character it is since all strings are unique and all sequences are unique from left to right.

One way to figure out what character you are reading is to put the characters into a tree. You start at the top of the tree, and for every bit you move either left for a 0, or right for a 1. When you complete a character, you start again at the top with the next bit. A tree for this example would be as shown in the chart to the left (the X's are just nodes).

Note: some strings require more bits with this compression scheme than if they were in their standard 8-bit ASCII form.

11. Tree Compression (cont.)

Input
The input file will contain an unknown number of lines, where each line contains a single ASCII string to compress.

Output
For each line of input, use the format described below to print the parts of the compressed string that would represent the binary form of the most compact storage of the string, either the Tree Compression form, or the standard ASCII string, whichever has the least amount of total bits.

The format for printing a compressed string is as follows:

- The first line of output is the binary representation of the byte (8 bits) that indicate the most compact storage form of the string. For a string in Tree Compression form, the 8 bits represent the number of unique characters in the compressed string. For a string in standard ASCII form, these bits will be all 0's.
- The remaining lines will be in one of these two formats:
  - If the string is in **compressed** form:
    - Print the byte (8 bits) representing the ASCII characters, one per line and in order of most used to least used. Characters that share the same count should be printed in the order of least to greatest by their ASCII order.
    - On the last line, print the string using the encoded values.
  - If the string is in **standard ASCII** form:
    - Print the binary representation of each character of the text string, one byte per line.
- Print a blank line after each set of output.

Note: For the example `ADA ATE APPLE`, the compressed form requires fewer bytes as shown below:
The **compressed** form would be (but your output will be one byte per line for the benefit of the judges):

```
00000111010000010010000001000101010100000100010001001100010101000
```

and the string using the encoded values would be:

```
01111001001111111101001110111011111011O
```

for a total of 104 bits.

The **ASCII** form would be (but your output will be one byte per line for the benefit of the judges):

```
0000000001000001010001000100000100100000010000010101010001000101
0010000001000001010100000101000001001100010001010
```

for a total of 112 bits (there is no carriage return, the bits are wrapped to the next line).

Example Input File

```
ADA ATE APPLE
ABCDEFG
ALFALFA
```

Example Output to Screen (cont. on next page)

11. Tree Compression (cont.)

Example Output to Screen (cont. on next page)

```
00000111
01000001
00100000
01000101
01010000
01000100
01001100
01010100
01111001001111111101001110111011111110110

00000000
01000001
01000010
01000011
01000100
01000101
01000110
01000111

00000011
01000001
01000110
01001100
01110011100
```

12. Words with Friends

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

The Facebook app, Words with Friends, takes the user's name as it is seen on Facebook and shortens it. In Facebook, a person's name reads first name, all middle names and last name with all names separated by a space. In Words with Friends, the first name is followed by the first initial of all of the middle and last names, all separated by a single space. For example, `Walter Alpheus Porter` in Facebook would become `Walter A P` in Words with Friends.

You are to write a program that will change a person's Facebook name to his Words with Friends name.

Input

The first line of input will contain a single integer `n` that indicates the number of names to follow. Each of the following `n` lines will contain a person's complete Facebook name. All names will contain only spaces and alphabetic characters.

Output

For each of the names input, you will print the person's Words with Friends name on a line by itself.

Example Input File

```
4
Richard S Jones
James Robin West
Alexander James Wilson
Mary Alice Jones Williams
```

Example Output to Screen

```
Richard S J
James R W
Alexander J W
Mary A J W
```

Note: Extra spaces at the end of each line of output are ok.