Greater New York Programming Contest
Kean University
Union, NJ

A • Mispelling$_4$

Misspelling is an art form that students seem to excel at. Write a program that removes the *n*th character from an input string.

Input

The first line of input contains a single integer ***N***, (1 ≤ ***N*** ≤ 1000) which is the number of datasets that follow.

Each dataset consists of a single line of input containing ***M***, a space, and a single word made up of uppercase letters only. ***M*** will be less than or equal to the length of the word. The length of the word is guaranteed to be less than or equal to 80.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the misspelled word. The misspelled word is the input word with the indicated character deleted.

Sample Input	**Sample Output**
`4`	`1 MISPELL`
`4 MISSPELL`	`2 ROGRAMMING`
`1 PROGRAMMING`	`3 CONTES`
`7 CONTEST`	`4 BALOON`
`3 BALLOON`	

B • Conversions

Conversion between the *metric* and *English* measurement systems is relatively simple. Often, it involves either multiplying or dividing by a constant. You must write a program that converts between the following units:

Type	**Metric**	**English equivalent**
Weight	`1.000 kilograms`	`2.2046 pounds`
	`0.4536 kilograms`	`1.0000 pound`
Volume	`1.0000 liter`	`0.2642 gallons`
	`3.7854 liters`	`1.0000 gallon`

Input

The first line of input contains a single integer ***N***, ($1 \le N \le 1000$) which is the number of datasets that follow.

Each dataset consists of a single line of input containing a floating point (double precision) number, a space and the *unit specification* for the measurement to be converted. The *unit specification* is one of **`kg`**, **`lb`**, **`l`**, or **`g`** referring to kilograms, pounds, liters and gallons respectively.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the appropriately converted value rounded to 4 decimal places, a space and the *unit specification* for the converted value.

Sample Input

```
5
1 kg
2 l
7 lb
3.5 g
0 l
```

Sample Output

```
1 2.2046 lb
2 0.5284 g
3 3.1752 kg
4 13.2489 l
5 0.0000 g
```

D • Decoding

Chip and Dale have devised an encryption method to hide their (written) text messages. They first agree secretly on two numbers that will be used as the number of rows (**R**) and columns (**C**) in a matrix. The sender encodes an intermediate format using the following rules:

1. The text is formed with uppercase letters [**A-Z**] and **<space>.**
2. Each text character will be represented by decimal values as follows:

   <space> = 0, **A** = 1, **B** = 2, **C** = 3, ..., **Y** = 25, **Z** = 26

The sender enters the 5 digit *binary* representation of the characters' values in a spiral pattern along the matrix as shown below. The matrix is padded out with zeroes (0) to fill the matrix completely. For example, if the text to encode is: "**ACM**" and **R=4** and **C=4**, the matrix would be filled in as follows:

$$\begin{tikzcd}
0 \arrow[r] & 0 \arrow[r] & 0 \arrow[r] & 0 \arrow[d] \\
1 \arrow[r] & 1 \arrow[r] & 0 \arrow[d] & 1 \arrow[d] \\
0 \arrow[u] & 0 & 1 \arrow[l] & 0 \arrow[d] \\
1 \arrow[u] & 1 \arrow[l] & 0 \arrow[l] & 0 \arrow[l]
\end{tikzcd}$$

A = 00001, C = 00011, M = 01101
(one extra 0)

The bits in the matrix are then concatenated together in *row major* order and sent to the receiver. The example above would be encoded as: 0000110100101100

This problem continues on the next page...

Input

The first line of input contains a single integer ***N***, ($1 \le N \le 1000$) which is the number of datasets that follow.

Each dataset consists of a single line of input containing **`R (1<=R<=20)`**, a space, **`C (1<=C<=20)`**, a space, and a string of binary digits that represents the contents of the matrix (**R * C** binary digits). The binary digits are in *row major* order.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the ***decoded*** text message. You should throw away any trailing spaces and/or partial characters found while decoding.

Sample Input	Sample Output
`4`	`1 ACM`
`4 4 0000110100101100`	`2 HI`
`5 2 0110000010`	`3 HI`
`2 6 010000001001`	`4 HI HO`
`5 5 0100001000011010110000010`	

E • Flipping Burned Pancakes

The cook at the *Frobbozz Magic Pancake House* sometimes falls asleep on the job while cooking pancakes. As a result, one side of a stack of pancakes is often burned. Clearly, it is bad business to serve visibly burned pancakes to the patrons. Before serving, the waitress will arrange the stacks of pancakes so that the burned sides are facing down. You must write a program to aid the waitress in stacking the pancakes correctly.

We start with a stack of ***N*** pancakes of distinct sizes, each of which is burned on one side. The problem is to convert the stack to one in which the pancakes are in size order with the smallest on the top and the largest on the bottom and burned side down for each pancake. To do this, we are allowed to flip the top ***k*** pancakes over as a unit (so the ***k-th*** pancake is now on top and the pancake previously on top is now in the ***k-th*** position and the burned side goes from top to bottom and *vice versa*).

For example (+ indicates burned bottom, - a burned top):

+1 -3 -2 [flip 2] $\Rightarrow$ +3 -1 -2 [flip 1] $\Rightarrow$ -3 -1 -2 [flip 3] $\Rightarrow$
+2 +1 +3 [flip 1] $\Rightarrow$ -2 +1 +3 [flip 2] $\Rightarrow$ -1 +2 +3 [flip 1] $\Rightarrow$ +1 +2 +3

You must write a program which finds a sequence of at most ***(3n – 2)*** flips, which converts a given stack of pancakes to a sorted stack with burned sides down.

Input

The first line of the input contains a single decimal integer, ***N***, the number of problem instances to follow. Each of the following ***N*** lines gives a separate dataset as a sequence of numbers separated by spaces. The first number on each line gives the number, ***M***, of pancakes in the data set. The remainder of the data set is the numbers ***1*** through ***M*** in some order, each with a plus or minus sign, giving the initial pancake stack. The numbers indicate the relative sizes of the pancakes and the signs indicate whether the burned side is up (-) or down (+). ***M*** will be, at most, **30**.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, the number of flips (***K,*** where ***K*** **>= 0**) required to sort the pancakes and a sequence of ***K*** numbers, each of which gives the number of pancakes to flip on the corresponding sorting step. There may be several correct solutions for some datasets. For instance **3 2 3** is also a solution to the first problem below.

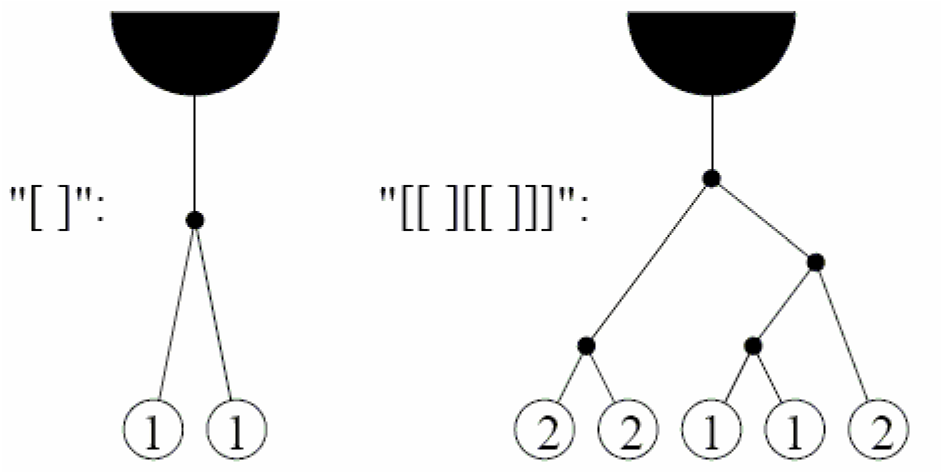
Sample Input	Sample Output
3	1 6 2 1 3 1 2 1
3 +1 -3 -2	2 6 4 1 4 3 1 2
4 -3 +1 -2 -4	3 3 5 1 5
5 +1 +2 +3 +4 -5	

F • Monkey Vines

Deep in the Amazon jungle, exceptionally tall trees grow that support a rich biosphere of figs and juniper bugs, which happen to be the culinary delight of brown monkeys.

Reaching the canopy of these trees requires the monkeys to perform careful navigation through the tall tree's fragile vine system. These vines operate like a see-saw: an unbalancing of weight at any vine junction would snap the vine from the tree, and the monkeys would plummet to the ground below. The monkeys have figured out that if they work together to keep the vines properly balanced, they can *all* feast on the figs and juniper bugs in the canopy of the trees.

A *vine junction* supports exactly two *sub-vines,* each of which must contain the same number of monkeys, or else the vine will break, leaving a pile of dead monkeys on the jungle ground. For purposes of this problem, a *vine junction* is denoted by a pair of matching square brackets **[ ]**, which may contain nested information about junctions further down its *sub-vines*. The nesting of vines will go no further than **25** levels deep.



You will write a program that calculates the *minimum* number of monkeys required to balance a particular vine configuration. There is **always** at least one monkey needed, and, multiple monkeys may hang from the same vine.

Input

The first line of input contains a single integer ***N***, (1 ≤ ***N*** ≤ 1000) which is the number of datasets that follow.

Each dataset consists of a single line of input containing a vine configuration consisting of a string of **[** and **]** characters as described above. The length of the string of **[** and **]** will be greater than or equal to zero, and less than or equal to 150.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the minimum number of monkeys required to reach the canopy successfully. Assume that all the hanging vines are reachable from the jungle floor, and that all monkeys jump on the vines at the same time.

Sample Input	**Sample Output**

`3`
`[]`
 	
`[[][[]]]`
	`1 2`
	`2 1`
	`3 8`

Note: The second line of sample input is a blank line.