Problem P: Partitioning containers

Our company is shipping cargo with total weight s. The shipping company prepared two ships. Both have a maximum capacity of $s/2$. Unfortunatelly our cargo is already packed into n containers of various weights. We are not able to open the containers, so we need to decide which containers should go on which ship.

That sounds as a hard problem (easpecially to computer scientists). Luckily, our insurance policy makes it possible to skip shipping *one* container.

Problem specification

You are given n positive integers with sum s. Erase at most one of them, and split the remaining ones into two groups. The sum of numbers in each group must not exceed $s/2$.

Input specification

The first line of the input file contains an integer t specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of two lines. The first line contains the number n of containers. The second line contains n positive integers: the weights of containers. The containers are labeled 1 through n in the order in which they appear in the input.

In the easy data set all test cases have $n \leq 20$. In the hard data set we have $n \leq 15\,000$.

Output specification

For each test case output two lines. Each line should describe the cargo of one ship. The first number in the line must be the number of containers in the ship's cargo. The rest of the line should contain the labels of all containers carried by the ship, in any order. Any valid solution will be accepted.

Example

input

```
2

4
10 20 30 71

4
10 10 10 10
```

output

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

First test case:
We have four containers with weights 10, 20, 30, and 71. Therefore $s = 10 + 20 + 30 + 71 = 131$, *and each ship will carry at most* $131/2 = 65.5$. *In the solution described above we put container #1 on the first ship, containers #2 and #3 on the second ship, and we throw away container #4. (Another valid solution is to put the first three containers on the same ship and to leave the other ship empty.)*

Second test case:
Note that we are not required *to throw away one of the containers. Sometimes we may be able to ship all of them. (But of course, there are other valid solutions that only ship three of the four containers.)*

Problem Q: Quaint encryption

Bob and Alice are constantly sending notes to each other during school lessons. As they sit on opposite sides of the classroom, the notes have to be handed over via several of their classmates.

One of those classmates is Mike. He is bored and wants to read the notes. Unfortunately for him, Bob and Alice are encrypting the messages, and he was unable to crack their code.

Then, one day Mike got a lucky break: Bob dropped a paper with decrypted words from one of Alice's messages. The words were not in their proper order, but this still helped Mike to find the encryption method. Afterwards, he could easily read all their messages.

Problem specification

Alice and Bob have a very simple encryption method: the substitution cipher, using just the lowercase letters of the English alphabet. The key to the cipher is some *permutation* of letters – a string that contains each of the 26 letters exactly once. For example, one such permutation is `ebmhcdfgijklnoapqrswtuvxzy`.

To encode a message, simply replace the i-th letter of the alphabet by the i-th letter of the key, for all i. For example, if we use the above key to encode a message, all '`a`'s will be encoded as '`e`'s, all '`b`'s as '`b`'s, all '`c`'s as '`m`'s, and so on. The word "`beef`" would then be encoded as "`bccd`".

You will be given a list of original words, and a list of the same words after they have been encrypted. The words in the second list are **not necessarily** in the same order as the words in the first list.

Your task is to produce any key that encodes the first set of words into the second set of words.

Input specification

The first line of the input file contains an integer t specifying the number of test cases. Each test case is preceded by a blank line.

Each test case starts with an integer n – the number of words in each list. Each of the following n lines contains one original word. Afterwards there are another n lines with the encrypted words. All words use lowercase English letters only. In the easy data set $n \leq 30$, in the hard data set $n \leq 10\,000$. No word has more than 30 characters.

Output specification

For each test case, output a single line containing a permutation of the 26 lowercase letters.

You may assume that for each test case at least one valid permutation exists. If there are multiple valid solutions, you may output any of them.

Example

input

```
1

2
dogg
cat
mew
haff
```

output

```
ebmhcdfgijklnoapqrswtuvxzy
```

In this case, the original words are `dogg` *and* `cat`*. Clearly, the encrypted form of* `dogg` *is* `haff` *and the encrypted form of* `cat` *is* `mew`*. This tells us that any permutation of the form* `e.mh..f.......a....w......` *will be a good encryption key. The example output contains one such permutation.*

Problem R: Rebel Alliance attacks

Oh no! The Rebel Alliance have found plans for a second Death Star – the deadliest weapon in the universe. It can destroy entire planets in a few seconds. If they allow it to be unleashed, it will mean a certain defeat.

The mission is clear – find Death Star and destroy it in a single big explosion. But now the Alliance needs to know how many explosives they need to prepare. And thus they need your help.

The Death Star can be represented as a perfect sphere with origin $(0, 0, 0)$ and radius r. Rebel scientists have determined that to ensure its destruction, the Rebels must infiltrate it and place explosives at all points (x, y, z) inside the sphere such that x, y, z are integers, and the distance between $(0, 0, 0)$ and (x, y, z) is also an integer.

Input specification

The first line of the input file contains an integer t specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consist of a single line containing a single integer r – the radius of the Death Star.

The largest test cases have $r = 100$ in the easy data set and $r = 44\,444$ in the hard data set.

Output specification

For each test case, output a single line containing a single integer – the number of explosives needed.

Example

input

```
3

1

2

5
```

output

```
1
7
49
```

In the first test case, the only explosive needs to be placed at the centre of the Death Star. In the second test case, additional explosives must be placed at positions $(\pm 1, 0, 0)$, $(0, \pm 1, 0)$, $(0, 0, \pm 1)$.