When printing text on paper we need ink. But not every character needs the same amount of ink to print: letters such as 'W', 'M' and '8' are more expensive than thinner letters as 'i', 'c' and '1'. In this problem we will evaluate the cost of printing numbers in several bases.

As you know, numbers can be expressed in several different bases. Well known bases are binary (base 2; digits 0 and 1), decimal (base 10; digits 0 to 9) and hexadecimal (base 16; digits 0 to 9 and letters A to F). For the general base n we will use the first n characters of the string "0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ", which means the highest base in this problem is 36. The lowest base is of course 2.

Every character from this string has an associated cost, represented by an integer value between 1 and 128. The cost to print a number in a certain base is the sum of the costs of all characters needed to represent that number. For the numbers given in the input, you will have to calculate the cheapest base(s) to represent this number in. Numbers in any base are printed without leading zeros.

Input

The input has less than 25 test cases. The first line of input file denotes this number of test cases. The description of every test case is given below:

The first 4 lines of every case contain 9 integers each: the costs of the 36 characters in the order given above. Then follows the number of queries on a line by itself. Every query appears on a line by itself and is formed by a number between 0 and 2000000000 in decimal format.

Output

For every case in the input, print one line '`Case` X`:`', without the quotes, where X is the case number starting from 1.

For every query within a case print one line '`Cheapest base(s) for number` Y`:`' followed by the cheapest base(s) in increasing order, separated by one space. Y is the query in decimal format. Print a blank line between cases.

The numbers in the second sample output are actually all in one line just like the first sample output. Due to lack of horizontal space they are shown broken in two lines.

Sample Input

```
2
10 8 12 13 15 13 13 16 9
11 18 24 21 23 23 23 13 15
17 33 21 23 27 26 27 19 4
22 18 30 30 24 16 26 21 21
5
98329921
12345
800348
14
873645
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1 1
4
0
1
10
100
```

Sample Output

```
Case 1:
Cheapest base(s) for number 98329921: 24
Cheapest base(s) for number 12345: 13 31
Cheapest base(s) for number 800348: 31
Cheapest base(s) for number 14: 13
Cheapest base(s) for number 873645: 22

Case 2:
Cheapest base(s) for number 0: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
Cheapest base(s) for number 1: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36
Cheapest base(s) for number 10: 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36
Cheapest base(s) for number 100: 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
26 27 28 29 30 31 32 33 34 35 36
```