4880 Scanning UPC Barcodes

The *UPC-A* bar code encodes 12 decimal digits in alternating "dark" and "light" bars as 15 patterns '`SLLLLLLMRRRRRRE`' where '`S`' is the start pattern '`101`' ('`1`' indicates "dark" and '`0`' indicates "light"), '`M`' is the middle pattern '`01010`' and '`E`' is the end pattern '`101`'. Each '`L`' is a left pattern corresponding to one of the first 6 digits and each '`R`' is a right pattern corresponding to one of the last 6 digits. The width of each bar is a multiple of a fixed value (the *X* dimension). Again a '`1`' indicates a "dark" band and '`0`' indicates a "light" band. The tick marks above the bar code illustration indicate the start of each code. There are $3 + 5 + 3 + 12 * 7 = 95$ bands total. In addition there must be at least 9 "light" bands at either end of the bar code.

0 4 9 7 0 5 6 8 2 3 0 2

digit	L pattern	R pattern
0	0001101	1110010
1	0011001	1100110
2	0010011	1101100
3	0111101	1000010
4	0100011	1011100
5	0110001	1001110
6	0101111	1010000
7	0111011	1000100
8	0110111	1001000
9	0001011	1110100

The last decimal digit in the code is a check sum digit which is computed as follows:

$$\begin{aligned} CheckSum &= 3 * (digit1 + digit3 + digit5 + digit7 + \\ &+ \ digit9 + digit11) + digit2 + digit4 + \\ &+ \ digit6 + digit8 + digit10. \\ Code &= CheckSumMod(10); \end{aligned}$$

If $Code = 0$, $check\ digit = 0$. Other wise, $check\ digit = (10 - Code)$.

A bar code scanner could use a camera to take a narrow image across the bar code and deduce the on/off pattern of bands as below:

```
10100011010100011000101101110110001101011000101010101000010010001101100100001011100101101100101
```

if the code was scanned right side up or the following if it was scanned upside down:

```
10100110110100111010000100110110001001000010101010100011010110001101110110100011000101011000101
```

Again, the tick marks above each image indicate the start of each code.

Unfortunately, the images are not always this clear due to lack of contrast or reflections off shiny material, as shown here:

When scanning the image, it is not always clear whether a particular band is dark or light. It is often still possible to determine the bar code even if we do not know exactly whether a particular band is "dark" or "light". First, only 20 of 128 possible 7-bit digit codes are used. Second, only codes with a correct check digit are valid. Finally, even if several codes match, it is unlikely that more than one will be in the database for a particular application. For this problem we will use a '`?`' to indicate uncertainty in the value of a particular band. The start (`S`), middle (`M`) and end (`E`) codes must match in order for a match to be considered valid.

Write a program which takes as input a string of 95 characters, '`0`', '`1`', or '`?`' and outputs all valid *UPC-A* digit strings which could scan to that sequence of band values in either direction.

Input

The first line of input contains a single integer P ($1 \leq P \leq 1000$), which is the number of data sets that follow. Each data set consists of 3 lines. The first line contains a single decimal integer which is the problem number (starting at 1). The second line contains the first 50 characters of the input string. The third line contains the final 45 characters of the input string. As noted above, the input string consists of only the characters '`0`', '`1`', or '`?`'.

Output

For each data set there are varying number of lines of output. If no UPC codes match the input string, then the only line of output should contain the problem number (starting at 1), a space, then the digit '`0`'. If more than 8 codes match the input string, the first line of output contains the problem number, a space, then the digit '`9`'. This line is followed by the first 8 codes which match, in ascending numeric order, one per line. Otherwise, the first line of output contains the problem number, a space, then a single decimal digit (`1..8`) which is the number of matching codes. This line is followed by the matching codes, in ascending numeric order, one per line.

Sample Input

```
3
1
10100110110100111010000100110110001001000010101010
100011010110001101110110100011000101011000101
2
1010001101010001100010????????????1101011000101010
101000010010001101100100001011100101101100101
3
1010001101010001100010????????????1101011000101010
101000010010001??????????01011100101101100101
```

Sample Output

```
1 1
049705682302
2 2
049705682302
049835682302
3 9
049005681302
049005682002
049035688302
049035689002
049105684302
```

```
049105685002
049135681302
049135682002
```