# 4698 Barcode of Judgment

As I continue to sit in traffic on I-35, I grow bored with calculating the substrings of car manufacturers and notice an ad for a particularly intriguing restaurant on a nearby cab. Noticing that the ad has an embedded 2-D barcode, I take out my cell phone, snap a picture, and my phone gives me a web address where I find a map to the nearest franchise location. (References: [1] Rekimoto, Jun and Ayatsuka, Yuji. 2000. CyberCode: Designing Augmented Reality Environments with Visual Tags. Sony Computer Science Laboratories, Inc.)

Unfortunately I'm still stuck in traffic. So instead of the delicious shrimp dish featured in the ad, I search my floorboards for a spare ketchup packet...

Given an "image" of a 2-D barcode, determine the text string that it encodes. 2-D barcodes have both fixed and variable parts. In the following diagram, the fixed parts are shown as "0" or "1", while variable parts are represented as "-":

```
1010---01
1000---00
10-------
10-------
10-------
1000---00
101000001
```

Note that the variable areas contain the data, and are decoded left-to-right, top-to-bottom when the barcode is oriented as shown. For example, given the actual barcode:

```
101000101
100011000
100100010
101101011
100111011
100011000
101000001
```

The data bits would be:

```
001110010001011010110111011110
```

To translate them into characters, we break them up into 5 bit chunks.

```
00111 00100 01011 01011 01110 11110
```

Then translate them into decimal.

```
7 8 11 11 14 30
```

Finally, decode them according to the following table:

| **Bit Value** | **Decoded Character** |
|---|---|
| 0–25 | Lowercase letters "a" through "z" |
| 26 | "_" (underscore) |
| 27 | "/" (forward slash) |
| 28 | ":" (colon) |
| 29 | "." (period) |
| 30 | "!" (exclamation point) |
| 31 | "?" (question mark) |

So the example barcode says:

```
hello!
```

## Input

The input will begin with a line containing a single integer $N$ ($1 \leq N \leq 100$) indicating the number of data sets. Each data set will begin with a line of the form '$H$ $W$' where $H$ ($7 \leq H \leq 80$) and $W$ ($9 \leq H \leq 80$) are respectively the height and width of the image in this data set. This is followed by $H$ lines of input, each of which contains $W$ binary digits (i.e., '0' or '1'). These bits represent the black & white images taken by your camera.

To simplify further, you can assume that any valid barcode photographed will be squarely photographed, though it may have been rotated any multiple of 90 degrees.

## Output

Each data set may contain a single, valid 2-D barcode, in which case you should decode it and display its value. It may contain no valid barcodes, in which case you should print 'NOCODE'. Finally, it may appear that there are multiple valid barcodes. In this case, other data in the image is obscuring your ability to identify the true barcode, so you should simply print 'INTERFERENCE'.

## Sample Input

```
3
10 10
0000000000
0101000101
0100011000
0100100010
0101101011
0100111011
0100011000
0101000001
0000000000
0000000000
10 10
0000000000
1010001010
1000110000
1001000100
1011010100
1001110100
1000110000
```

```
1010000010
0000000000
0011111111
10 10
0000000000
0101000101
0100011000
0100100010
0001101011
0100111011
0100011000
0101000001
0000000000
0000000000
```

## Sample Output

```
hello!
INTERFERENCE
NOCODE
```