Hamming Code

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

Jaeha is writing an operating system for his toys, so they will be able to communicate with each other. However, the wireless chips in his toys are very cheap, so they are prone to transmission errors. Quite frequently, Jaeha is seeing some noises in the transmitted data: some bits get flipped during the transmission process. Jaeha wants to implement Hamming Code to remedy this situation.

The following is a brief description of how Hamming Code works.

Hamming(7,4) code encodes 4 bits of data into a 7-bit code, by adding 3 bits of parity data. The parity data ensures that the receiver will be able to decode the correct data even when one of the 7 bits get flipped during transmission. The encoding process is as follows.

1. First, we number the bits in the encoded message from 1 to 7.
2. Next, we assign the data to be transmitted to the 3rd, 5th, 6th, 7th bit of the encoded data. For example, if we were to transmit data 1011, the encoded message will look like _ _ 1 _ 0 1 1, leaving 3 bits to be filled in.
3. Next, the remaining bits are filled by parity data, from the first bit, as follows.
   - The 1st bit is filled so that the XOR of 1st, 3rd, 5th and 7th bit is 0. In the above example, the XOR of 3rd, 5th and 7th bit is already 0, so the 1st bit must be 0.
   - The 2nd bit is filled so that the XOR of 2nd, 3rd, 6th and 7th bit is 0. In the above example, the XOR of 3rd, 6th and 7th bit is 1, so the 2nd bit must be 1.
   - The 4th bit is filled so that the XOR of 4th, 5th, 6th and 7th bit is 0. In the above example, the XOR of 5th, 6th and 7th bit is already 0, so the 4th bit must be 0.
4. After the process, we get the fully encoded message 0 1 1 0 0 1 1.

```
Position
1 2 3 4 5 6 7
Raw Message
_ _ 1 _ 0 1 1
Encoded Message
0 1 1 0 0 1 1
```

Now, let's talk about the decoding process of Hamming(7,4) code. To see how the code corrects an isolated error, let's suppose that the 3rd bit gets flipped incorrectly when the message 0 1 1 0 0 1 1 is transmitted. Therefore, the receiver will receive the following corrupted message instead: 0 1 0 0 0 1 1.

```
Position
1 2 3 4 5 6 7
Encoded Message
0 1 1 0 0 1 1
Error
_ _ x _ _ _ _
Corrupted Message
0 1 0 0 0 1 1
```

The receiver then calculates a 3-bit integer, called syndrome, as follows.

The 1st (lowest) bit is determined as the XOR of 1st, 3rd, 5th and 7th bit. In the above corrupted message, this bit will be 1. The 2nd bit is determined as the XOR of 2nd, 3rd, 6th and 7th bit. In the

above corrupted message, this bit will be 1. The 3rd bit is determined as the XOR of 4th, 5th, 6th and 7th bit. In the above corrupted message, this bit will be 0. Concatenating these three digits, the syndrome will be $011_2 = 3_{10}$: the location of the error bit.

```
Position
1 2 3 4 5 6 7 XOR
Check 1
0 _ 0 _ 0 _ 1 1
Check 2
_ 1 0 _ _ 1 1 1
Check 3
_ _ _ 0 0 1 1 0
```

syndrome $= 011_2 = 3_{10}$

Thus the receiver can flip the 3rd bit in the received message. The resulting sequence of 0's and 1's will be now correct and we can take the 3rd, 5th, 6th and 7th bit to get the original message: 1 0 1 1.

```
Position
1 2 3 4 5 6 7
Corrupted Message
0 1 0 0 0 1 1
Indicated Error
_ _ x _ _ _ _
Corrected Message
0 1 1 0 0 1 1
Decoded Message
_ _ 1 _ 0 1 1
```

Note when all bits are transmitted correctly, the syndrome will be 0 and no bit needs to be flipped.

As a smart baby, Jaeha had no problem writing the encoding procedure. However, his parents don't allow him to use the computer more than 10 minutes a day, because he is only 3 months old. Let's help Jaeha by implementing the decoding procedure

Input

The input consists of T test cases. The number of test cases T is given in the first line of the input. The next T line each contains a transmitted message in 7 binary digits, starting from the 1st bit. The message will contain at most a single incorrect bit.

Output

For each test case, you must output a single line of four binary digits denoting the decoded message.

Example

standard input	standard output
2 0100011 1111111	1011 1111