Problem D

Number Game

Source: `numbergame.(c|cc|pas|java)`
Input: `read from stdin`

Christine and Matt are playing an exciting game they just invented: the Number Game. The rules of this game are as follows.

The players take turns choosing integers greater than 1. First, Christine chooses a number, then Matt chooses a number, then Christine again, and so on. The following rules restrict how new numbers may be chosen by the two players:

- A number which has already been selected by Christine or Matt, or a multiple of such a number, cannot be chosen.
- A sum of such multiples cannot be chosen, either.

If a player cannot choose any new number according to these rules, then that player loses the game.

Here is an example: Christine starts by choosing 4. This prevents Matt from choosing 4, 8, 12, etc. Let's assume that his move is 3. Now the numbers 3, 6, 9, etc. are excluded, too; furthermore, numbers like: $7 = 3+4, 10 = 2\cdot 3+4, 11 = 3+2\cdot 4, 13 = 3\cdot 3+4, \ldots$ are also not available. So, in fact, the only numbers left are 2 and 5. Christine now selects 2. Since $5 = 2+3$ is now forbidden, she wins because there is no number left for Matt to choose.

Your task is to write a program which will help play (and win!) the Number Game. Of course, there might be an infinite number of choices for a player, so it may not be easy to find the best move among these possibilities. But after playing for some time, the number of remaining choices becomes finite, and that is the point where your program can help. Given a game position (a list of numbers which are not yet forbidden), your program should output all *winning moves*.

A winning move is a move by which the player who is about to move can force a win, no matter what the other player will do afterwards. More formally, a winning move can be defined as follows.

- A winning move is a move after which the game position is a losing position.
- A winning position is a position in which a winning move exists. A losing position is a position in which no winning move exists.
- In particular, the position in which all numbers are forbidden is a losing position. (This makes sense since the player who would have to move in that case loses the game.)

Input

The input file consists of several test cases. Each test case is given by exactly one line describing one position.

Each line will start with a number n ($1 \le n \le 20$), the number of integers which are still available. The remainder of this line contains the list of these numbers $a_1, \ldots, a_n$ ($2 \le a_i \le 20$).

The positions described in this way will always be positions which can really occur in the actual Number Game. For example, if 3 is not in the list of allowed numbers, 6 is not in the list, either.

At the end of the input file, there will be a line containing only a zero (instead of n); this line should not be processed.

Output

For each test case, your program should output "`Test case #`*m*", where *m* is the number of the test case (starting with 1). Follow this by either "`There's no winning move.`" if this is true for the position described in the input file, or "`The winning moves are:` w_1 w_2 ... w_k" where the w_i are all winning moves in this position, satisfying $w_i < w_{i+1}$ for $1 \leq i < k$. After this line, output a blank line.

Sample Input

```
2 2 5
2 2 3
5 2 3 4 5 6
0
```

Sample Output

```
Test Case #1
The winning moves are: 2

Test Case #2
There's no winning move.

Test Case #3
The winning moves are: 4 5 6
```