Hi-Q is a popular solitaire game that comes in a small box with a playing board that has little holes in the shape of a cross and 32 little pegs that fit into the holes. Starting with the centermost hole open, players move the pegs by jumping one peg over another, either in a horizontal or vertical direction and removing each peg that is jumped over. Diagonal jumps are not allowed. The object for players is to remove as many pegs from the board as possible. This problem involves writing a program that will automatically play Hi-Q so that we can investigate how the game might unfold based on various opening arrangements of pegs.

There is a peg board with the following shape and with holes numbered from 1 to 33 as follows:

```
                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
```

An instance of the game begins with some holes having pegs in them and the rest of the holes being empty. The game proceeds by jumping one peg over another, either horizontally or vertically, with the peg that is jumping landing in an empty hole, and the peg being jumped over being removed from the board. For example, if 9 is empty and 10 and 11 are not, then the peg in 11 can be "moved" to 9 with the peg in 10 being removed. After this move, 10 and 11 would both be empty but 9 would have a peg in it.

Given a specific board configuration your program will pick and model a specific move, over and over, until no more moves are available. Your program will then report the sum of the holes that still have pegs in them. At any point during the game there may be more than one possible move available. In such a case always model the move with the target hole of the moving peg as large as possible. If there is more than one move available to the largest possible target hole, then choose from those moves the one with the larger source hole.

For example, if the board looks like this, with '`X`' representing a peg and '`O`' representing a hole:

```
                O   O   O
                O   O   O
 O   O  O  X  O  X  O
 O   O  O  X  O  X  O
 O   O  O  O  X  O  O
                O   O   O
                O   O   O
```

then the following jumps would be made:

1.- from 12 over 19 to 26 (26, 24, and 5 are the only possible targets and 26 is the largest),

2.- from 25 to 27 over 26 (note that 26 is a peg after the jump 1, and 5, 24, 27 the only possible targets being 27 the largest)

3.- from 10 to 24 over 17 (5 and 24 are the only possible targets with $24 > 5$) 17 to 29 ($29 > 5$), and

Two pegs would be left, one in hole 24 and one in hole 27. Thus 51 would be reported as the result for this instance.

Input

The first line contains an integer N between 1 and 10 describing how many instances of the game are represented. The remaining lines will describe N instances of the game by listing the holes which begin with pegs in them, in increasing order. A '`0`' will indicate the end of each sequence of unique numbers between 1 and 33 that represents an instance of the game.

Output

There should be $N + 2$ lines of output. The first line of output will read '`HI Q OUTPUT`'. There will then be one line of output for each instance of the game, reporting the sum of the holes that still have pegs in them for the final configuration of that instance. The final line of output should read '`END OF OUTPUT`'.

Sample Input

```
4
10 12 17 19 25 0 0 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 0
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 18 19 20
21 22 23 24 25 26 27 28 29 30 31 32 33 0
```

Sample Output

```
HI Q OUTPUT
51
0
561
98
END OF OUTPUT
```