1997 Spring Denison Computer Programming Contest
Problem 1
A Toy

There is a child's toy that consists of a number of small plastic squares with clips on their edges so that you can fasten two of them together either in the same plane (making a 1-by-2 square, in effect) or at right angles. By fastening many of these together you can make little boxes. For example, if you have 10 of these little squares, you can fasten them together to make a 1-by-1-by-2 box (4 sides are 1-by-2 and the remaining two sides are 1-by-1). If you have 24 of these squares, you can make a 2-by-2-by-2 box (with all six sides being 2-by-2), among others. Note we are only interested in boxes that are "closed"; that is, no open ends are allowed.

The question here is if you have a pile of n of these little squares, what are the dimensions of the box with the largest volume you can make with these squares. Note you may indeed have squares left over after building this largest box. For example, if $n = 30$ you can make a 2-by-2-by-2 box of volume 8 using 24 squares, or a 1-by-1-by-7 box of volume 7 using all 30 squares. Hence, using all the squares does not necessarily lead to a box of largest volume.

Input for this problem will be a series of integers, one per line, each representing the number of squares. These integers will be no greater than 50,000. For each line of input you are to generate one line of output with 3 integers, separated by a single space, giving the dimensions of the largest box you can build using only these squares. The last input line will be zero and should generate no output. *Warning: Some care must be taken in choosing the dimensions to try. If your method is inefficient, it will not run within the required time.*

Example

On input:

```
100
1000
30
0
```

the output should look like:

```
4 4 4
11 12 16
1 3 3
```

1997 Spring Denison Computer Programming Contest
Problem 2
A Solitaire Game

There is a solitaire game called "four corners". To play this game, you deal out 4 cards, face up, in four piles (numbered 1 through 4, dealt in that order). You then discard any card showing that is of the same suit, but lower denomination, as another card showing. (In this game the cards run from ace (lowest) to king (highest).) After removing all possible cards, you deal 4 more cards on the top of these four piles and discard as before. Notice, however, that discarding a card may expose one beneath it, which may lead to another discard. In the later stages of the game, this discarding may go on for some time. This continues until all 52 cards are dealt. You win if either all 4 kings are on the top, or you have only one pile left. (In practice, this is a very rare event.)

Input for this program will be 5 sets of 52 integers, the integers in each set are numbered 0 through 51. Each set will start on a new line, is "free formatted", and may run many lines. Each integer represents a card in the standard playing deck: 0 through 12 represent the ace through king of clubs, 13 through 25 represent the ace through king of diamonds, 26 through 38 represent the ace through king of hearts and 39 through 51 represent the ace through king of spades.

For each data set, you are to produce one line of output giving the number of cards in each pile 1, 2, 3 and 4 in that order, separated by a single space. If you have a winning configuration at the end, also print an appropriate message indicating so.

Example

On input:

```
16 50 4 34 28 31 46 41 0 25 30 20 49 23 7 32 27 29 8 44
18 3 22 35 51 38 45 17 19 2 37 14 9 42 10 33 39 24 40
43 21 1 48 15 5 6 13 12 11 47 26 36
44 40 28 32 23 3 22 33 20 13 17 8 6 14 4 5 47
10 26 35 36 43 27 31 9 2 19 11 42 37 34 0 49 16 41 7
18 24 45 39 48 21 46 30 1 50 15 29 12 25 38 51
21 46 9 48 24 31 50 20 19 23 44 6 15 40 26 13 0 37 34
25 22 5 41 30 12 49 16 43 28 1 14 36 18 35 4 45 8 38
42 29 10 17 7 27 39 51 47 32 33 11 2 3
17 46 50 38 14 11 5 19 31 33 0 26 6 18 7 35 45 34 2 20
37 30 13 12 4 8 47 44 39 48 32 29 40 25 42 41 3 10 9
27 36 16 22 1 43 49 21 51 15 28 23 24
11 45 22 35 38 30 8 18 2 34 37 24 21 44 16 6 48 32 1
12 47 20 41 33 10 15 23 7 26 0 5 4 25 49 50 27 40 14
42 29 46 13 9 36 19 43 28 39 31 17 51 3
```

the output should look like:

```
2 8 5 2
10 7 2 9  A WIN
10 10 0 5
3 4 0 7
11 3 8 9
```

1997 Spring Denison Computer Programming Contest
Problem 3
Ranking

Everyone loves to rank things: teams, movies, ball players, beers; you name it. In this problem, you are presented with someone's ranking of (unspecified) items along with a second party's critique of that ranking. The second party's critique comments on the ranking of each item. For each item, the second party either agrees on ranking (that is, the item should stay in the same position in the ranking), thinks the item should be ranked higher (that is, come earlier in the ranking) or should be ranked lower (that is, come later in the ranking). Your job is to provide a new ranking consistent with the second party's critique, if possible, or determine this adjustment is impossible.

Input will consist of multiple input sets. The first line of the input set will be an integer n (< 100). If n is zero, input terminates. There will then follow n items ranked and critiqued, one per line. Each line will have an item name, which will be a string of no more than 10 non-white space characters letter, followed by a single space and either the letter “a”, “u” or “d”, indicating “agree”, the ranking should move “up” (that is, ranked earlier) or the ranking should move ”down” (that is, ranked later).

For each input set, print a message on a line, as shown below, indicating the number of the input set, followed by a modified ranking or a message saying a new ranking is impossible, starting on the next line. The modified ranking should be a listing of the items one per line with a single space separating item letters. A blank line should separate output from different input sets. Note that there may be many possible rerankings consistent with the critique; you may print any one of these.

Example

On input:

```
6
A a
B d
D a
E u
F d
H u
5
Betty d
Cecil u
Dave u
Ernie u
Fred u
4
Apples d
Brownies d
Cookies a
Cake u
5
Fred d
Betty d
George u
John d
Phil a
0
```

the output should look like:

```
For input set 1:
A
E
D
B
H
F

For input set 2:
Cecil
Dave
Ernie
Fred
Betty

For input set 3:
Cake
Apples
Cookies
Brownies

For input set 4:
No reranking is possible.
```

1997 Spring Denison Computer Programming Contest
Problem 4
A Factory Problem

A factory produces Gizmos in a two-step process. The factory receives Whatsits and processes them in "type A" machines to get an intermediate Whosit. A "type B" machine then takes a Whosit and does the final processing to produce the finished Gizmo. Once a type A machine produces a Whosit, it is immediately available to a type B machine. The factory may have many type A machines and many type B machines, which work in parallel. Unfortunately, the time each machine takes to finish its work varies; some machines are older and take longer than the newer models. Your job is to determine the earliest completion time for producing a bunch of Gizmos from a given number of Whatsits, when given the number of machines of each type and their job times.

There will be multiple input sets for this program. Each input set will consist of 4 lines. The first line will contain the number of Whatsits the factory starts with in the morning. (If this number is zero, input terminates.) The next line is of the form $n\ a_1\ a_2 \ldots a_n$, where n (> 0 but < 10) is the number of type A machines and $a_1 \ldots a_n$ are the job times, in minutes, for the n type A machines. These will all be positive integers. The third line is similar to the second line and gives the corresponding information about the type B machines. The fourth line is blank.

For each input set, produce one line of output indicating the earliest time all the Whatsits are processed into Gizmos, assuming all Whatsits are available at time 0. This time should be in the form `h:mm` where `h` is the number of hours and `mm` is the number of minutes. Note that none of these positions should be left blank. You are guaranteed that all Gizmos will be processed within 8 hours.

Example

On input:

```
4
1 20
2 40 30

100
5 1 1 1 1 1
9 1 8 1 3 1 4 1 1 10

30
3 10 5 20
2 20 15

0
```

the output should look like:

```
1:50
0:21
4:30
```

1997 Spring Denison Computer Programming Contest
Problem 5
Number Triangles

A number triangle is a group of positive integers of the form below.

```
        3
      5   6
    1   2   4
  8   2   5   2
1   5   2   1   3
```

You are to find the value of the path from the top of the triangle to the bottom row that is the largest. The value of a path is the sum of the numbers on it. A path moves from top to bottom by moving one row down either right diagonally or left diagonally. For example, the path in the above triangle that has theh largest value is 3-5-1-8-5 with a value of 22.

Input will be a series of number triangles. The first line will be an integer $n < 100$ indicating the number of rows in the triangle; a zero indicates the end of input. There will then follow n rows of integers giving the rows of the triangle from top to bottom.

For each input set, you should produce one output line consisting of a single integer, giving the largest value of a path from the top to the bottom of the triangle. **Warning:** *Do not try to produce all paths from the top of the triangle to the bottom; this will exceed your time limit.*

Example

On input:

```
5
3
5 6
1 2 4
8 2 5 2
1 5 2 1 3
3
100
43 2
1 3 70
10
3
2 4
5 2 3
2 4 3 1
5 2 1 5 7
6 2 9 2 2 1
54 5 3 2 1 5 60
2 3 4 2 2 2 2 1
3 2 2 1 2 100 2 3 1
3 2 1 2 4 5 3 2 2 5
0
```

the output should look like:

```
22
172
134
```