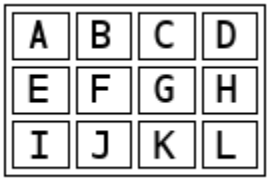
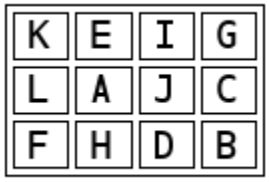
Problem A : Arrangement

Given a 3 x 4 grid of distinct letters. The letters in the grid are rearranged so that no letter remains in the same row or in the same column. In addition it is known that, in the new arrangement, certain given sets of letters appear in the same row or in the same column. You are required to write a program to determine the new arrangement of letters in the grid, assuming that there exists such a unique arrangement.

As an illustration, consider the arrangement given on the left. Assume that after rearrangement of letters no letter remains in the same row or in the same column. In addition, assume that letters in each string of letters: LAJ, KIG, HDB, ACJ and EIG appear in the same row while the pair of letters appearing in each string: KL, AH, ID, GB, JD and LF appears in the same column. On the basis of this information the program should find the new arrangement shown on the right.





Input

The input may contain multiple test cases.

Each test case contains three lines.

The first line gives a sequence of three strings of distinct letters, each of length four. The j^{th} letter in the i^{th} string (i=1, 2, 3; j=1, 2, 3, 4) represents the letter in j^{th} column and i^{th} row of the grid. The second line gives a sequence of strings of distinct letters, each of length three. The letters in each string appear in the same row, after rearrangement. The third line gives a sequence of strings of distinct letters, each of length two. The letters in each string appear in the same column, after rearrangement.

A blank character separates two consecutive strings in a line.

The input terminates with an input line containing 0 for a test case.

Output

For each test case, print in one line, a sequence of three strings of distinct letters, each of length four. The j^{th} letter in the i^{th} string (i=1, 2, 3; j=1, 2, 3, 4) represents the letter in j^{th} column and i^{th} row of the grid after rearrangement.

Use a blank character to separates two consecutive strings in a line.

Sample Input (A.in)

```
ABCD EFGH IJKL
LAJ KIG HDB ACJ EIG
KL AH ID GB JD LF
0
```

Sample Output (A.out)

```
KEIG LAJC FHDB
```

Problem B : Walker

Mr. Walker is a person who is known for his ability to walk fast. He accepts an attractive offer to get as much free land as he can cover by a walk on an open uneven huge piece of land within a specified time and following certain conditions.

A walk is a sequence of paths, starting at a marked spot on the land and ending at the same spot where the walk starts. The first path in a walk begins at the marked spot and extends in one of the four directions: North (N), South (S), East (E) or West (W). It ends at the point where the direction changes.

Each of the other paths in a walk begins at the point where the previous path ends and ends either at a point where the direction changes or when the walk ends. Mr. Walker may change directions, as and when he feels like, depending on the constraint of time and/or quality/quantity of the land he decides to get. When he changes a direction he must keep the direction always to one of the four directions N, S, E or W. Paths are distinct and nonintersecting; for example, a path in E or W direction cannot cross another in N or S direction.

Given a walk, you are required to write a program that finds the area of land covered by the walk.

Input

The input consists of multiple test cases.

For each test case there is only one input line. The line gives a walk defined by a sequence of paths. A direction followed, without any space, by a distance represents a path. A direction is denoted by one of the four letters N, S, E or W while a distance is measured in meters and is denoted by an integer. A space character appears between two paths in the given sequence.

The input terminates when a line containing 0 appears as a test case.

Output

For each test case print the area of land covered by the given walk.

Sample Input (B.in)

```
N3 W4 S8 E4 N5
W6 N2 E9 S6 W3 N4
W6 N2 W3 S4 W5 S4 E14 N6
0
```

Sample Output (B.out)

```
32
30
80
```

Problem C : Delete

Delete the least number of integers from a given set of integers so that the product of the remaining integers in the set is a perfect square. In case there is more than one solution then find the solution that gives the largest perfect square. Assume that each integer contains five or less number of digits. The total number of integers in the given set is twenty or less. You are required to write a program for a problem as simple as this.

Input

The input may contain multiple test cases.

For each test case there is a single input line. The line contains the given set of integers. The input terminates with a line containing 0 as input.

Output

For each test case there is only one output line. The line simply prints the integers to be deleted in ascending order. There are two special cases; print output for these cases as indicated below.

Case 1: No integer is to be deleted: Print 0 as output.

Case 2: All integers are to be deleted: Print all integers in ascending order.

Sample Input (C.in)

```
2    3  12  18  24
12  10  15  18
4   12  10  15
10  12  15
0
```

Sample Output (C.out)

```
24
0
10  12  15
10  12  15
```

Problem D : Maximum Score

Professor Anupam Shukla is fond of playing with matrix. One day in his class he has defined a matrix which is filled up with scores but these scores can be positive, negative or zero. Professor Shukla has called the position of an element in a matrix by cell. Every cell is connected to its right, left, top and bottom only if the corresponding cell exists. Score of a connected component is the sum of scores in the cells of the connected component.

Professor Shukla has asked his students to write a program to find a connected component of cells from the matrix that has the highest aggregate score. If there are two or more connected components with the same maximal score, return the one with the largest size, i.e., the one with the largest number of cells.

Input

The input may contain multiple test cases.

First line of the input of the program will be the dimensions of the matrix and the scores are given on the next line onwards.

Output

The output should be the total score followed by the matrix showing the cells in the connected component. Other cells should be represented by 'x'.

Sample Input (D.in)

```
 3  4
-1  4 -6  7
 3  2 -9 -9
-3  0 -5  4
```

Sample Output (D.out)

```
10
X 4 -6 7
3 2  X X
X 0  X X
```

Problem E : Numerology

Suppose there are some groups of friends who all believe fervently in numerology. They decide to choose the best group among them by aligning the vowels of their names. In order to account for mismatches in the number of vowels, a special symbol '-' is introduced that can augment the total number of vowels to make them the same. A symmetric matrix of size 6x6 is conceived to encode the scores of matching each vowel plus the gap to every other vowel and the gap. Even though the scores can be positive and negative, certain constraints apply.

For example, the score of a vowel with itself is positive, while that of a gap with any character (including another gap) is negative. When aligning their names, the column-wise scores are accumulated. For each column, the score is the cumulative of the pair-wise scores. For example, if the alignment of vowels of three names AEI, AO and EU is

```
AEI
A-O
EU
```

then the score for the first column is score(A,A) + score(A,-) + score(-,A), while that of the second and third columns are score(E,-) + score(-,E) + score(E,E) and score(I,O) + score(O,U) + score(U,I) respectively. The total score is the sum of the scores of the three columns.

Write an algorithm to compute the maximum possible score of a group of friends. Assume that there are 3 to 5 friends in a group.

Input

The input may contain multiple test cases.

The first line contains the information about the number of friends in the group. If the number is *n*, then the next *n* lines contain the names of these *n* friends. Next six lines provide the 6x6 score matrix ordered according to A, E, I, O, U,-. For example, score (I, O) is the value in the 3rd column of the 4th row. The input terminates with a line containing 0 as input.

Output

Output the maximum total score followed by the corresponding alignments.

Sample Input (E.in)

```
3
SATTEKI DRY
ANMOL WRY
SEMI FRY
+3 +1 +1 −1 −2 −4
+1 +5 +2 −3 −3 −4
+1 +2 +3 +0 −1 −4
−1 −3 +0 +3 +2 −4
−2 −3 −1 +2 +4 −4
−4 −4 −4 −4 −4 −4
0
```

Sample Output (E.out)

```
−2
AEI
-AO
-EI
```

Problem F : Tree

Assume, in a forest there are n trees and i^{th} tree is at position i, for i =1 ...n. These trees are having different heights. Let us assume that the tree i is having nonnegative heights h_i feet. One day the forester has decided to trim the heights or uproot the trees to arrange the rooted trees with respect to their heights. That is, at the end of the operation, rooted trees satisfy the property of $h_i < h_j$ for $i < j$. Note that if one uproots a tree, then it cannot be placed back in any place and hence it is no longer in the forest.

The problem is to determine is the minimum cutting of wood required to make sure that all remaining trees satisfy the height property.

Input

The input may contain multiple test cases.

Each test case contains a sequence of n integers to indicate the heights of n trees in the forest. The i^{th} integer indicates the height of the i^{th} tree.

Output

For each test case, output is the total length of the wood cut. It is followed by a line which tells the amount of cut from each tree, i.e., i^{th} element in the line tells the amount of cut in the i^{th} tree. Next line contains the information about the final height of each tree which means that i^{th} element in that line tells the current height of the i^{th} tree. If a tree is uprooted, then mark its final height as 'x'.

Sample Input (F.in)

```
2 5 1 2 7
```

Sample Output (F.out)

```
3
0 0 1 2 0
2 5 x x 7
```

Problem G : Badness

Assume that there are *n* vectors each of size 2*k*. Number the dimensions of the vector as 1, 2, …, 2*k*. Define the badness of a vector by the sum of the absolute values of the *k* differences of the *k* consecutive pairs.

For example, the badness of the vector {1, 4, 3, 2} is |1 4| + |3 2| = 4.

The badness of the set of *n* vectors is the cumulative badness of the elements. However, the dimensions can be shuffled to produce another vector.

For example, the above vector can be shuffled to produce {2, 1, 4, 3} which improves the badness to 2.

The problem is to choose a shuffling of the vectors such that the cumulative badness is minimized. Note that the shuffling must be consistent and must be applied to all the vectors in the set, i.e., if two columns are interchanged, they must be interchanged for all the vectors.

Input

The input may contain multiple test cases.

First line of a test case contains two information. First one is the number of vectors while second one tells about the length of the vector. If the number of vectors is *n*, each of the length say, *2k*, then next lines provide the vectors. The input terminates with a line containing 0 as input.

Output

Output the total badness value followed by the shuffled vectors.

Sample Input (G.in)

```
2  4
1  5  4  2
2  2  3  3
```

Sample Output (G.out)

```
4
2  1  5  4
3  2  2  3
```