Postman

(Running time - 1s.)

In a countryside a postman has to deliver post to customers that live in villages and along every road connecting the villages.

Your task is to help the postman design a route that goes though every village and every road at least once--the input data are such that this is always possible. However, each route also has a cost. The people in the villages wish that the postman visit their village as early as possible. Therefore, each village has made the following deal with the post: If the village *i* is visited as the *k*-th \textbfdifferent village on the tour and *k<=w(i)*, the village pays *w(i)-k* euros to the post. However, if *k>w(i)*, the post agrees to pay *k-w(i)* euros to the village. Moreover, the post pays the postman one euro for each road on the tour.

There are *n* villages, numbered form 1 to *n*. The post is located in the village number one, so the route should start and end in this village. Each village is placed on the crossing of two roads, on the crossing of four roads, or there is a road going through the village (i.e. there are 2, 4, or 8 roads going out of each village). There can be several roads connecting the same villages or a road can be a loop, i.e. connect a village with itself.

Task

Your task is to write a program that:

- reads the description of villages and connecting them roads, from the input file `pos.in`,
- designs such a route that goes through each village and road at least once and maximizes the total profit (or minimizes the loss) of the post,
- writes the result to the output file `pos.out`.

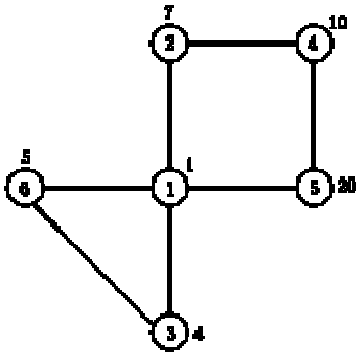
If there are several possible solutions, your program should output just one of them.

Input

In the first line of the input file `pos.in`, there are two integers *n* and *m*, separated by a single space; *n*, *1<=n<=200*, is the number of villages and *m* is the number of roads. In each of the following *n* lines there is one positive integer. The *i+1*-th line contains *w(i)*, *1<=w(i)<=1 000*, specification of the fee paid by the village number *i*. In each of the following *m* lines there are two positive integers separated by a single space--villages connected by consecutive roads.

Output

Your program should write one positive integer *k*, the length of the route, to the first line of the output file `pos.out`. The following line should contain *k+1* numbers of consecutive villages on the route $v_1,v_2,...,v_{k+1}$, separated by single spaces, with $v_1=v_{k+1}=1$.



Example

Input file `pos.in`:

```
6 7

1

7

4

10

20

5

2 4

1 5

2 1

4 5

3 6

1 6

1 3
```

correct answer in file `pos.out`

```
7

1 5 4 2 1 6 3 1
```

Crack the Code

Cryptography is the science and technology of coding messages so that only the intended recipient can read them. Cryptanalysis, on the other hand, is the science of breaking the codes. For this problem, assume you're a cryptanalyst hired to break open a series of encrypted messages captured in a police raid to the headquarters of the local mafia. Your colleagues have already reverse engineered the encryption program (see `crack.pas` or `crack.c` for the code), and the only thing left to do is to reverse the algorithm and guess the key used to encrypt the files. Along with the encrypted files, there are also some plaintext files that are believed to originate from the same source as the encrypted files and thus have similar structure in terms of language, word usage, etc.

Task

Your task is to decrypt the given messages and save them in the specified files. You do not have to provide any program--just the decrypted messages.

Input

You are given several data sets. Set consists of the files `cra`*n*`.*`, where *n* is the number identifying the data set. Each data set consists of the files:

- `cra*.in`, encrypted message,
- `cra*.txt`, plaintext files of the same origin, as the encrypted message.

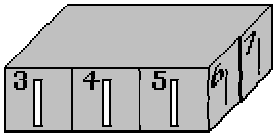
Output

For each encrypted message `cra*.in`, you should save the decrypted message in the file `cra*.out`.

Box of Mirrors

(Running time - 1s.)

Mathematician Andris likes different puzzles and one of his favorites is covered box of mirrors. If we look at the horizontal cross-section of such a box, we can see that its basement contains *n*m* square cells (*n* rows, and *m* columns). In each cell there can be placed mirror which is oriented diagonally from lower left corner to upper right corner. Both sides of the mirror reflect light. At the box edges opposite to each cell row or column there is a gap through which you can light a beam into box or the beam can come out of the box. Through each gap you can light the beam in only one direction--perpendicular to the edge containing the gap. Therefore, beam reflecting from mirror changes its direction by 90°. When the beam goes through empty cells, its direction doesn't change. Gaps are numbered consecutively from 1 to *2*(n+m)*, around the box, counter-clockwise, starting from the gap on the left side of the upper left cell and going downwards. Since arrangement of mirrors in the box is not visible, the only way to determine it is by lighting beams in some gaps and watching where light comes out.



Task

Write program that:

- reads the size of the box and gaps describing beams coming in and out of the box from the input file `box.in`,
- determines in which cells there are mirrors and which cells are empty,
- writes the result to the output file `box.out`.

If there are several possible solutions, your program should output anyone of them.

Input

First line of input file `box.in` contains two positive integers: *n* (the number of cells rows, *1<=n<=100*) and *m* (the number of cells columns, *1<=m<=100*) separated by a single space. Each of the following *2*(n+m)* lines contains one positive integer. The number in the *i+1*-th line denotes the number of gap from which light comes out if it is lightened into gap number *i*.

Output

Your program should write to the output file `box.out` *n* lines, each of them containing *m* integers separated by single spaces. The *j*-th number in the *i*-th line should be 1, if there is a mirror in the cell in the *i*-th row and *j*-th column of the box, or it should be 0 if the cell is empty.

Example

Input file `box.in`:

```
2 3

9

7

10

8

6

5

2

4

1

3
```

correct answer in file `box.out`

```
0 1 0

0 1 1
```