#Problem 1: Complex Coin Change

Description:
You will be given N coins and an amount K. You can use each coin an infinite number of times. You have to print the minimum number of coins needed to make the amount K.

In the first line you will be given N and K. In the following line, you will be given N values denoting N coins.

Limits
1<=N<=30
1<=K<=10000

Test Cases:

Input	Output
3 7 1 2 3	3
11 30 1 13 20 23 36 37 38 51 61 94 97	6
11 73 1 13 20 23 36 37 38 51 61 94 97	2
11 500 1 13 20 23 36 37 38 51 61 94 97	6

#Problem 2: Complex Coin Change with Coin Print

Description:
You will be given N coins and an amount K. You can use each coin an infinite number of times. You have to make the amount K using the minimum number of coins. You also have to print which coins have been taken with their usage count in ascending order over the coin's value aka smaller valued coin will be printed first.

In the first line, you will be given N and K. In the following line, you will be given N values denoting N coins.
For each input, print the coins with their usage count in ascending order. If there exists multiple solutions print any of them. Print each coin's output in separate lines. Look at the input output section for more clarification.

Limits

1<=N<=30
1<=K<=10000

Test Cases:

Input	Output
3 7 1 2 3	1 1 3 2
11 30 1 13 20 23 36 37 38 51 61 94 97	1 4 13 2
11 73 1 13 20 23 36 37 38 51 61 94 97	36 1 37 1

11 500
1 13 20 23 36 37 38 51 61 94 97
	51 1
	61 1
	97 4

#Problem 3: 0/1 Knapsack

Description:
You will be given N items and a knapsack weight W. Each item N[i] has two elements, its total positive benefit b[i] and its total weight w[i]. You have to choose the elements in such a way that you can maximize your total benefit not exceeding W. You can not take a fractional amount of weight for an item.

In the first line, you will be given N and W. In the first following line you will be given N values, each element denotes the total positive benefit where i^{th} value is for the i^{th} element. In the second following line, you will again be given N values, each denoting the weights where i^{th} value denotes the total weight for i^{th} element.

For each input, you have to output the maximum positive benefit you can achieve not exceeding the knapsack weight.

Limits
1<=N<=30
1<=W<=1000

Test Cases:

Input	Output

4 10
2 3 5 7
2 3 5 7
	10

3 30
25 10 30
10 20 30
	35

10 37
86 87 24 87 15 5 87 96 76 16
14 31 6 56 43 69 33 39 50 24
	111

10 65
86 87 24 87 15 5 87 96 76 16
14 31 6 56 43 69 33 39 50 24
	206

10 29
86 87 24 87 15 5 87 96 76 16
14 31 6 56 43 69 33 39 50 24
	110

#Problem 4: LCS and Path Print

Description:
Given two strings M and N. You need to print the longest common subsequence (LCS) length found between M and N. You also need to print a lcs which exists between M and N.

In the first line, you will be given M and in the second line you will be given N. M and N will contain only digits or English alphabets.

As output, in the first line print the lcs length. In the second line, print a lcs which exists between M and N. If there exists multiple solutions print any of them. Look at the input output section for more clarification.

Limits
1<=|M|<=40
1<=|N|<=40
|M|, |N| denote the length of string M and N respectively.

Test Cases:

Input	Output
AAAB AAAB	4 AAAB
ACAAB	5

ACDAAB	ACAAB
ECBEBECABB BACACABECAEBEBA	7 BBECABB
EBEEA AAECC	1 A
EECECABBBC BBCCBEACAEEB	5 CECAB
CECCBCCECAACABEAABBCCBCECCBCEECECAE ABEACCBBACCEAEEBCABECAAAAAE	16 ABEACCBBCEECECAE

#Problem 5: LIS, Longest Increasing Subsequence

Description:
You will be given an array of size N having N integer numbers. You need to calculate the longest increasing subsequence length of the array.

In the first line you will be given N. In the following line you will have N integer numbers.

Limits
1<=N<=30

Test Cases:

Input	Output
3 1 2 3	3
5 1 100 2 3 4	4
7 1 100 2 105 7 109 111	5
10 9 2 5 3 7 11 8 10 13 6	6
15 46 33 66 34 65 4 28 32 41 70 51 2 35 11 26	5

#Problem 6: Hill Climbing

Description:
An avenger is trying to climb a very dangerous hill. There are some places on the hill which are very steep and dangerous to climb. Also, there are some places on the hill which are pretty flat and possess no difficulties.

This hill climbing problem can be modeled as a 2D grid problem. Where the grid has M rows and N columns. Analogically here, M denotes the height and N denotes the width of the hill respectively.

The climber is at the bottom now. He will start climbing the hill aka start traversing the grid from the bottom row. In the grid, for each cell a value is given which denotes the danger lying in that cell. If the climber comes in this cell, it will add danger for him in the path to climb the hill. In this problem, you need to find the optimal path for the climber to reach the top of the hill, **minimizing** the total danger of climbing. From each cell, a climber can make three possible moves which has been shown in the following figure,

**climbed complete hill*

Can go here	Can go here	Can go here	
	Current cell		

**hill bottom*

In the first line, you will be given two values M and N denoting the number of rows and columns of the grid. Then you will have M lines of values where each line will have N values. Here the values lying in the ith line denote the values of the ith row of the grid. In such ith line, jth value denotes the danger value of grid cell (i,j).

In the output, you need to print the minimum possible danger value for the climber to reach the top row. Outside of the bottom row means, he did not start climbing and outside of the top row means, he has finished climbing. The climber can not move right from any cells belonging to the rightmost column, similarly can not move left from the leftmost column, because it would bring death to him.

Limits
1<=M,N<=40
0<=Grid cell [i,j] <=1000 for all (i,j) pairs

Test Cases:

Input	Output

3 3
2 10 0
5 0 7
0 6 4
	0

4 3
0 2 1
9 8 2
3 2 7
10 5 2
	7

3 3
14 25 0
1 5 10
5 1 2
	6

3 4
1 4 4 4
4 4 1 4
2 1 4 3
	6

10 10
12 96 71 28 53 50 24 83 99 30
73 70 37 97 99 1 32 99 84 13
73 13 15 49 69 50 96 3 55 17
32 77 47 41 51 75 3 80 44 87
16 20 2 64 20 53 97 54 17 93
14 7 99 72 66 4 4 84 4 28
47 57 56 12 86 65 10 19 59 99
51 61 31 96 7 5 31 37 86 6
78 37 55 86 99 11 72 13 17 51
87 86 32 11 52 1 78 1 34 76
	146