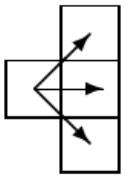
116 Unidirectional TSP

Problems that require minimum paths through some domain appear in many different areas of computer science. For example, one of the constraints in VLSI routing problems is minimizing wire length. The Traveling Salesperson Problem (TSP) — finding whether all the cities in a salesperson's route can be visited exactly once with a specified limit on travel time — is one of the canonical examples of an NP-complete problem; solutions appear to require an inordinate amount of time to generate, but are simple to check.

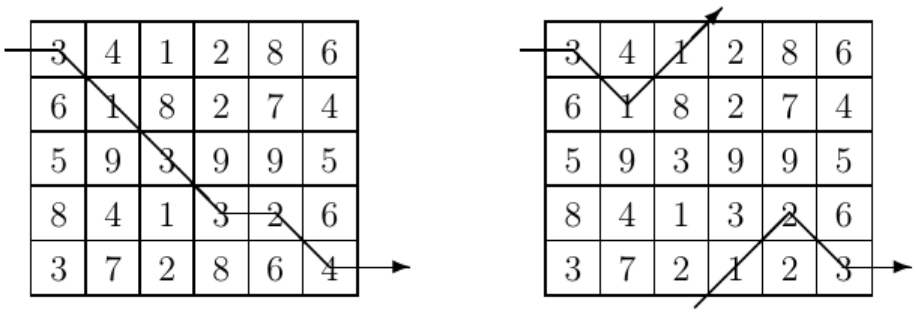
This problem deals with finding a minimal path through a grid of points while traveling only from left to right.

Given an $m \times n$ matrix of integers, you are to write a program that computes a path of minimal weight. A path starts anywhere in column 1 (the first column) and consists of a sequence of steps terminating in column n (the last column). A step consists of traveling from column i to column $i+1$ in an adjacent (horizontal or diagonal) row. The first and last rows (rows 1 and m) of a matrix are considered adjacent, i.e., the matrix "wraps" so that it represents a horizontal cylinder. Legal steps are illustrated on the right.



The *weight* of a path is the sum of the integers in each of the n cells of the matrix that are visited.

For example, two slightly different 5×6 matrices are shown below (the only difference is the numbers in the bottom row).



The minimal path is illustrated for each matrix. Note that the path for the matrix on the right takes advantage of the adjacency property of the first and last rows.

Input

The input consists of a sequence of matrix specifications. Each matrix specification consists of the row and column dimensions in that order on a line followed by $m \cdot n$ integers where m is the row dimension and n is the column dimension. The integers appear in the input in row major order, i.e., the first n integers constitute the first row of the matrix, the second n integers constitute the second row and so on. The integers on a line will be separated from other integers by one or more spaces. **Note:** integers are not restricted to being positive.

There will be one or more matrix specifications in an input file. Input is terminated by end-of-file.

For each specification the number of rows will be between 1 and 10 inclusive; the number of columns will be between 1 and 100 inclusive. No path's weight will exceed integer values representable using 30 bits.

Output

Two lines should be output for each matrix specification in the input file, the first line represents a minimal-weight path, and the second line is the cost of a minimal path. The path consists of a sequence of n integers (separated by one or more spaces) representing the rows that constitute the minimal path. If there is more than one path of minimal weight the path that is *lexicographically* smallest should be output.

Note: *Lexicographically* means the natural order on sequences induced by the order on their elements.

Sample Input

```
5 6
3 4 1 2 8 6
6 1 8 2 7 4
5 9 3 9 9 5
8 4 1 3 2 6
3 7 2 8 6 4
5 6
3 4 1 2 8 6
6 1 8 2 7 4
5 9 3 9 9 5
8 4 1 3 2 6
3 7 2 1 2 3
2 2
9 10 9 10
```

Sample Output

```
1 2 3 4 4 5
16
1 2 1 5 4 5
11
1 1
19
```

UVa Online Judge

231 Testing the CATCHER

A military contractor for the Department of Defense has just completed a series of preliminary tests for a new defensive missile called the CATCHER which is capable of intercepting multiple incoming offensive missiles. The CATCHER is supposed to be a remarkable defensive missile. It can move forward, laterally, and downward at very fast speeds, and it can intercept an offensive missile without being damaged. But it does have one major flaw. Although it can be fired to reach any initial elevation, it has no power to move higher than the last missile that it has intercepted.

The tests which the contractor completed were computer simulations of battlefield and hostile attack conditions. Since they were only preliminary, the simulations tested only the CATCHER's vertical movement capability. In each simulation, the CATCHER was fired at a sequence of offensive missiles which were incoming at fixed time intervals. The only information available to the CATCHER for each incoming missile was its height at the point it could be intercepted and where it appeared in the sequence of missiles. Each incoming missile for a test run is represented in the sequence only once.

The result of each test is reported as the sequence of incoming missiles and the total number of those missiles that are intercepted by the CATCHER in that test.

The General Accounting Office wants to be sure that the simulation test results submitted by the military contractor are attainable, given the constraints of the CATCHER. You must write a program that takes input data representing the pattern of incoming missiles for several different tests and outputs the maximum numbers of missiles that the CATCHER can intercept for those tests. For any incoming missile in a test, the CATCHER is able to intercept it if and only if it satisfies one of these two conditions:

1. The incoming missile is the first missile to be intercepted in this test.

-or-

2. The missile was fired after the last missile that was intercepted and it is not higher than the last missile which was intercepted.

Input

The input data for any test consists of a sequence of one or more non-negative integers, all of which are less than or equal to 32,767, representing the heights of the incoming missiles (the test pattern). The last number in each sequence is `-1`, which signifies the end of data for that particular test and is not considered to represent a missile height. The end of data for the entire input is the number `-1` as the first value in a test; it is not considered to be a separate test.

Output

Output for each test consists of a test number (`Test #1`, `Test #2`, etc.) and the maximum number of incoming missiles that the CATCHER could possibly intercept for the test. That maximum number appears after an identifying message. There must be at least one blank line between output for successive data sets.

Note: The number of missiles for any given test is not limited. If your solution is based on an inefficient algorithm, it **may not** execute in the allotted time.

Sample Input

```
389
207
155
300
299
170
158
65
-1
23
34
21
-1
-1
```

Sample Output

```
Test #1:
  maximum possible interceptions: 6

Test #2:
  maximum possible interceptions: 2
```

507 Jill Rides Again

Jill likes to ride her bicycle, but since the pretty city of Greenhills where she lives has grown, Jill often uses the excellent public bus system for part of her journey. She has a folding bicycle which she carries with her when she uses the bus for the first part of her trip. When the bus reaches some pleasant part of the city, Jill gets off and rides her bicycle. She follows the bus route until she reaches her destination or she comes to a part of the city she does not like. In the latter event she will board the bus to finish her trip.

Through years of experience, Jill has rated each road on an integer scale of "niceness." Positive niceness values indicate roads Jill likes; negative values are used for roads she does not like. There are not zero values. Jill plans where to leave the bus and start bicycling, as well as where to stop bicycling and re-join the bus, so that the sum of niceness values of the roads she bicycles on is maximized. This means that she will sometimes cycle along a road she does not like, provided that it joins up two other parts of her journey involving roads she likes enough to compensate. It may be that no part of the route is suitable for cycling so that Jill takes the bus for its entire route. Conversely, it may be that the whole route is so nice Jill will not use the bus at all.

Since there are many different bus routes, each with several stops at which Jill could leave or enter the bus, she feels that a computer program could help her identify the best part to cycle for each bus route.

Input

The input file contains information on several bus routes. The first line of the file is a single integer b representing the number of route descriptions in the file. The identifier for each route (r) is the sequence number within the data file, $1 \leq r \leq b$. Each route description begins with the number of stops on the route: an integer s, $2 \leq s \leq 20,000$ on a line by itself. The number of stops is followed by $s-1$ lines, each line i ($1 \leq i < s$) is an integer n_i representing Jill's assessment of the niceness of the road between the two stops i and $i+1$.

Output

For each route r in the input file, your program should identify the beginning bus stop i and the ending bus stop j that identify the segment of the route which yields the maximal sum of niceness, $m = n_i + n_{i+1} + \ldots + n_{j-1}$. If more than one segment is maximally nice, choose the one with the longest cycle ride (largest $j - i$). To break ties in longest maximal segments, choose the segment that begins with the earliest stop (lowest i). For each route r in the input file, print a line in the form:

`The nicest part of route` r `is between stops` i `and` j

However, if the maximal sum is not positive, your program should print:

`Route` r `has no nice parts`

Sample Input

```
3
3
  -1
   6
```

```
10
   4
  -5
   4
  -3
   4
   4
  -4
   4
  -5
4
  -2
  -3
  -4
```

Sample Output

```
The nicest part of route 1 is between stops 2 and 3
The nicest part of route 2 is between stops 3 and 9
Route 3 has no nice parts
```

531 Compromise

In a few months the European Currency Union will become a reality. However, to join the club, the Maastricht criteria must be fulfilled, and this is not a trivial task for the countries (maybe except for Luxembourg). To enforce that Germany will fulfill the criteria, our government has so many wonderful options (raise taxes, sell stocks, revalue the gold reserves,...) that it is really hard to choose what to do.

Therefore the German government requires a program for the following task:

> Two politicians each enter their proposal of what to do. The computer then outputs the longest common subsequence of words that occurs in both proposals. As you can see, this is a totally fair compromise (after all, a common sequence of words is something what both people have in mind).

Your country needs this program, so your job is to write it for us.

Input

The input file will contain several test cases.

Each test case consists of two texts. Each text is given as a sequence of lower-case words, separated by whitespace, but with no punctuation. Words will be less than 30 characters long. Both texts will contain less than 100 words and there will be at least one common word in each text. Each text will be terminated by a line containing a single '#'.

Input is terminated by end of file.

Output

For each test case, print the longest common subsequence of words occuring in the two texts. If there is more than one such sequence, any one is acceptable. Separate the words by one blank. After the last word, output a newline character.

Sample Input

```
die einkommen der landwirte
sind fuer die abgeordneten ein buch mit sieben siegeln
um dem abzuhelfen
muessen dringend alle subventionsgesetze verbessert werden
#
die steuern auf vermoegen und einkommen
sollten nach meinung der abgeordneten
nachdruecklich erhoben werden
dazu muessen die kontrollbefugnisse der finanzbehoerden
dringend verbessert werden
#
```

Sample Output

```
die einkommen der abgeordneten muessen dringend verbessert werden
```

UVa Online Judge

674 Coin Change

Suppose there are 5 types of coins: 50-cent, 25-cent, 10-cent, 5-cent, and 1-cent. We want to make changes with these coins for a given amount of money.

For example, if we have 11 cents, then we can make changes with one 10-cent coin and one 1-cent coin, two 5-cent coins and one 1-cent coin, one 5-cent coin and six 1-cent coins, or eleven 1-cent coins. So there are four ways of making changes for 11 cents with the above coins. Note that we count that there is one way of making change for zero cent.

Write a program to find the total number of different ways of making changes for any amount of money in cents. Your program should be able to handle up to 7489 cents.

Input

The input file contains any number of lines, each one consisting of a number for the amount of money in cents.

Output

For each input line, output a line containing the number of different ways of making changes with the above 5 types of coins.

Sample Input

```
11
26
```

Sample Output

```
4
13
```

990 Diving for gold

John is a diver and a treasure hunter. He has just found the location of a pirate ship full of treasures. The sofisticated sonar system on board his ship allows him to identify the location, depth and quantity of gold in each suken treasure. Unfortunatelly, John forgot to bring a GPS device and the chances of ever finding this location again are very slim so he has to grab the gold now. To make the situation worse, John has only one compressed air bottle.

John wants to dive with the compressed air bottle to recover as much gold as possible from the wreck. Write a program John can use to select which treasures he should pick to maximize the quantity of gold recovered.

The problem has the following restrictions:

- There are n treasures $\{(d_1, v_1), (d_2, v_2), \ldots (d_n, v_n)\}$ represented by the pair (depth, quantity of gold). There are at most 30 treasures.
- The air bottle only allows for t seconds under water. t is at most 1000 seconds.
- In each dive, John can bring the maximum of one treasure at a time.
- The descent time is $td_i = w * d_i$, where w is an integer constant.
- The ascent time is $ta_i = 2w * d_i$, where w is an integer constant.
- Due to instrument limitations, all parameters are integer.

Input

The input to this program consists of a sequence of integer values. Input contains several test cases. The first line of each dataset should contain the values t and w. The second line contains the number of treasures. Each of the following lines contains the depth d_i and quantity of gold v_i of a different treasure.

A blank line separates each test case.

Note:

In this sample input, the bottle of compressed air has a capacity of 200 seconds, the constant w has the value 4 and there are 3 treasures, the first one at a depth of 10 meters and worth 5 coins of gold, the second one at a depth of 10 meters and worth 1 coin of gold, and the third one at 7 meters and worth 2 coins of gold.

Output

The first line of the output for each dataset should contain the maximum amount of gold that John can recover from the wreck. The second line should contain the number of recovered treasures. Each of the following lines should contain the depth and amount of gold of each recovered treasure. Treasures should be presented in the same order as the input file.

Print a blank line between outputs for different datasets.

Sample Input

```
210 4
3
10 5
10 1
7 2
```

Sample Output

```
7
2
10 5
7 2
```