TIUP 2011

Torneio Inter-Universitário de Programação
Quarta Prova
http://tiup.ubi.pt

Universidade da Beira Interior
Departamento de Informática

September, 21^{th}
17h:00-20:00

Scientific Committee

Paul Crocker	crocker@di.ubi.pt
Pedro Inácio	inacio@di.ubi.pt
Pedro Patrício	pedrofp@ubi.pt
Fernando Pereira	fmtp@ubi.pt
Manuela Pereira	mpereira@di.ubi.pt
Hugo Proença	hugomcp@di.ubi.pt
Simão Melo de Sousa	desousa@di.ubi.pt

General Information

1. The contest follows ICPC rules (to know more, go to http://icpc.baylor.edu/).
2. It has a duration of 3 hours for the 6 problems.
3. The programs must read from the standard-input and write to the standard-output.
4. The generated test inputs use the following norms:
   - There are no spaces at the end of the lines and each line ends with a carriage return.
   - No multiple spacing is being used, unless it is explicitly mentioned on the problem description.
5. The generated test outputs follow the same rules.
6. All problems have deterministic answers (that is, given an input, there is only one possible output) except when the oposite is *explicitely* mentionned in the problem statement. In the case of multiple possible outputs, the evaluation system is able to approve all possible positive answers and reject the others.
7. All problems have 1 second timeout.
8. Each team (3 persons maximum) should use one PC and it is strictly forbidden to use any online resource other than Mooshak or what is given in the local PC disk.
9. Further information can be obtained by clicking on "Help" on your Mooshak account screen.
10. Please ask your questions to the jury using the "Questions" button.

Dialect	Compiler	Version	Command line	Extension
C	gcc	4.6.0	`gcc -ainsi -Wall -lm $file`	.c
C++	g++	4.6.0	`g++ -Wall $file`	.cpp
Java	OpenJDK	1.6.0_22	`javac $file`	.java
Pascal	fpc	2.4.4	`fpc -v0w -oprog $file`	.pas
Perl	perl	5.12.4	`perl $file`	.pl
Python	python	2.7.1	`python $file`	.py
OCaml	ocaml	3.12	`ocamlopt unix.cmxa str.cmxa nums.cmxa $file`	.ml

Note: Programs producing warnings or errors during compilation will be automatically rejected by the evaluation system.

Problem A: UDC, Undeterministic Data Compression

Suppose someone has proposed a (bit naive) data compression strategy, which has an undesirable property: it is non-deterministic, meaning that each sequence can be compressed in different ways and yield different results. This compression method works only for natural numbers and the compression strategy is quite simple: sequences of a symbol are replaced by the cardinality (the number of repeated adjacent symbols, up to 9) plus the respective symbol.
Take, for instance the raw sequence 1111. This compression process can yield the compressed sequence 41 (that means 4 times the symbol "1"). The raw sequence 000112225 can be compressed to 30213215.

Task

According to the afore description, most sequences can be compressed in different ways. For instance the 1111111111 sequence can be compressed in several different ways: 9111, 1191, 8121, 613111,...

The problem here is to simply calculate the number of different compressed sequences that may result from a given raw sequence.

Input

The input consists of a one line long uncompressed sequence of symbols (0-9).

Output

The output should be the total number of different compressed sequences that may result from the original sequence.

Constraints

The length of the input sequence is at most 60.

Sample Input

```
11112222
```

Sample Output

```
64
```

Sample Input 2

```
123456789
```

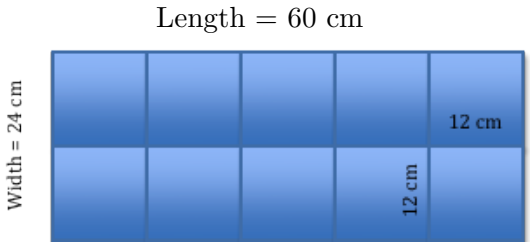
Sample Output 2

```
1
```

Problem B: SQUARES

Mr. Square has finally found a very good job (easy work and a good salary). The employer needs someone for the difficult task of dividing rectangular lands into equal sized square lands. The problem is that he wants the squared lands to be the biggest possible. In the opinion of the employer it is a very difficult task and he pays a very good salary. Mr. Square accepted the job immediately!

Example:



Task

Given the dimension of the rectangular land, your task is to output length of the side of the square.

Input

Two integers (L and W) separated by a space. They represent the Length and the Width of the rectangular land.

Output

An integer representing the length of the side of the square.

Constraints

$0 < L < 2^{32}$
$0 < W < 2^{32}$

Input example

```
60 24
```

Output example

```
12
```

Input example

```
100532645 3519
```

Output example

```
17
```

Problem C: Syncronization Sequence for 10GEPONs

During the standardization phase of the 10 Gigabit Ethernet Passive Optical Networks (10GEPON) technology, there was a need to find a so-called *syncronization sequence* (a series of 0s and 1s) with n bits (for simplicity, assume that n is an even number), identifying the beginning of each Ethernet frame. This sequence had to possess an important property:

the Hamming distance between the sequence and all $n-1$ ***shifted versions of itself had to be larger than, or equal to,*** $n/2-1$.

See below for a better explanation of what this means exactly.

Task

Consider that sequence s_1 and sequence s_2 have the same number of bits: n (where n is an even number). Consider also that $H(s_1, s_2)$ represents the Hamming distance between s_1 and s_2, i.e. $H(s_1, s_2)$ is the number of 1s in the sequence $s_3 = s_1 \oplus s_2$, where $\oplus$ denotes the bitwise xor operation. According to what was written in the previous paragraph, a sequence s is a suitable *syncronization sequence* if

$$H(s, s >>> i) \geq n/2 - 1,$$

for all $0 < i \leq n$. Notice that, in the previous expression and from now on, $>>>$ denotes the bitwise shift operation where the most significant (higher order) bit is filled with a 1 each time the sequence is shifted to the right.

Example:
Consider that $n = 4$, and $s =$ `1001`.
For $i = 1$, `1001` $\oplus$ `1100` $=$ `0101`.
Therefore,

$$H(s, s >>> 1) = 2.$$

For $i = 2$, `1001` $\oplus$ `1110` $=$ `0111`.
Therefore,

$$H(s, s >>> 2) = 3.$$

For $i = 3$, `1001` $\oplus$ `1111` $=$ `0110`.
Therefore,

$$H(s, s >>> 3) = 2.$$

Hence, the sequence s = 1001 would be a suitable *syncronization sequence*, since all the Hamming distances are larger than $4/2 - 1 = 1$.

Your mission, should you decide to accept it, is to implement a program that receives the number of bits of the sequence as input (an even number larger or equal than 4 and smaller or equal than 128), and returns a sequence with the aforementioned properties **in time to win this competition**.

Input

The input is a positive even number n larger or equal than 4 and smaller or equal than 128 (i.e. $n \in 2\mathbb{N}, 4 \leq n \leq 128$).

Output

The output should be a sequence of 0s and 1s with lenght equal to n.
Notice that there may be more than one suitable sequence for each input provided. Your program just needs to output **one** of those possible sequences, since the evaluation system includes a routine to test if it complies with the condition.

Constraints

For this contest, assume that $n \in 2\mathbb{N}$ and $4 \leq n \leq 128$.

Input example 1

```
64
```

Output example 1

```
0001000000101000001010110110110001011000100111001100001101010111
```

Input example 2

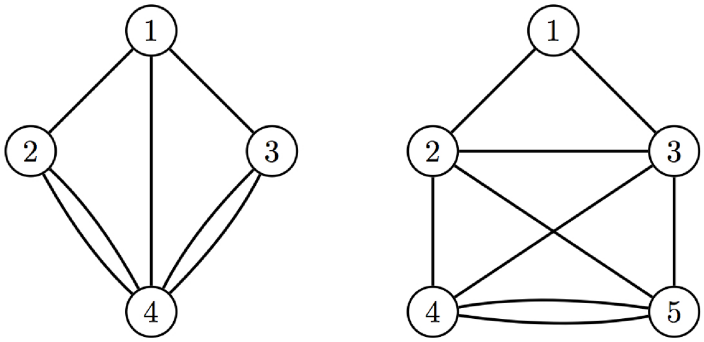
```
66
```

Output example 2

```
000001000100100101000100011110111101000001100011010001111101011011
```

Problem D: Eulerian Graphs

In Graph Theory, a multigraph is a graph that can have several edges between the same pair of nodes. Given an undirected connected multigraph $G = (V, E)$, where $V = \{1, \ldots, n\}$ denotes the set of nodes and E the set of edges, an *Eulerian trail* is a trail which visits every edge exactly once. An *Eulerian cycle* is an Eulerian trail which starts and ends on the same node. If such a cycle exists, the multigraph is called an *Eulerian graph*. The following figure represents two multigraphs: a non Eulerian graph (G, a.k.a. the Königsberg bridges graph) and an Eulerian graph (G'). The sequence of nodes $1, 2, 3, 4, 5, 2, 4, 5, 3, 1$ on G' is an Eulerian cycle.



$G = (V, E)$ $\qquad$ $G' = (V, E')$

The (symmetric) adjacency matrix of an undirected multigraph with $|V| = n$ is the $n \times n$ matrix $A = [a_{ij}]$, where a_{ij} is the number of edges between nodes i and j. The adjacency matrix which represents G' (illustrated in the figure above) is:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ & 0 & 1 & 1 & 1 \\ & & 0 & 1 & 1 \\ & & & 0 & 2 \\ & & & & 0 \end{bmatrix}$$

As a matter of fact, in this problem we are only interested in multigraphs whose adjacency matrix has the main diagonal filled with 0's. Such multigraphs are said to be loopless (there are no reflexive edges).

Task

Given an undirected loopless connected multigraph $G = (V, E)$, your goal is to determine whether G is Eulerian or not and, if the former holds, output an Eulerian cycle.
Hint: A well known result from Graph Theory states that an undirected loopless connected multigraph can be decomposed into edge-disjoint cycles if and only if all of its nodes have even degree.

Input

The input consists of n lines. The first contains one integer, representing the number of nodes ($n = |V|$). The remaining $n - 1$ lines introduce the adjacency matrix of G without the main diagonal, where the i^{th} line contains $(n - i)$ integers (single space separated).

Output

Alternatively:

- One line with the word `No` if G is non Eulerian;

  or

- Two lines. The first line with the word `Yes` if G is Eulerian, followed, in the second line, by a sequence of $(|E| + 1)$ node numbers (single space separated), describing an Eulerian cycle (if G is Eulerian). Note that there may exist more than one solution. The evaluation system is able to recognize all solutions, hence your task is to output **one** of them.

Constraints

$2 \leq |V| \leq 100$ and $1 \leq |E| \leq 7000$.

Input example 1 (G)

```
4
1 1 1
0 2
2
```

Output example 1 (G)

```
No
```

Input example 2 (G')

```
5
1 1 0 0
1 1 1
1 1
2
```

Output example 2 (G')

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

Problem E: GRAPH PARTITION

Consider a connected graph $G = (V, E)$, where $V = \{1, 2, \ldots, i, \ldots, n\}$ denotes the set of vertices and $E = \{e_1, e_2, \ldots, e_k, \ldots, e_m\} \subset V \times V$ denotes the set of edges. A partition, $Y = \{y_1, y_2, \ldots, y_K\}$, of G is defined as a partition of V, where each y_u represent a connected subgraph of G.

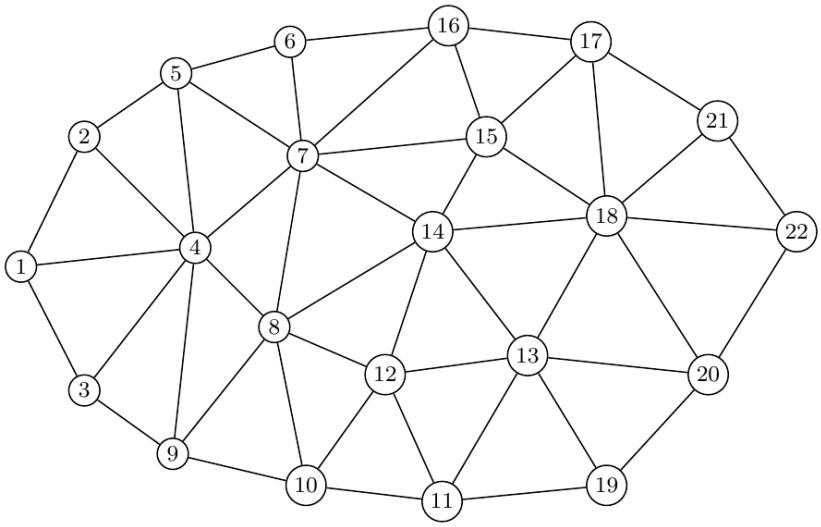
Task

Given a partition $Y = \{y_1, y_2, \ldots, y_K\}$ of G and a set of non overlapping connected subgraphs $Y' = \{y'_1, y'_2, \ldots, y'_{K'}\}$ your task is to compute a new partition composed of subgraphs Y' along with the equivalence classes of the equivalence relation $\backsim$ defined on $V \setminus \bigcup_{j=1}^{K'} y'_j$ as follows:

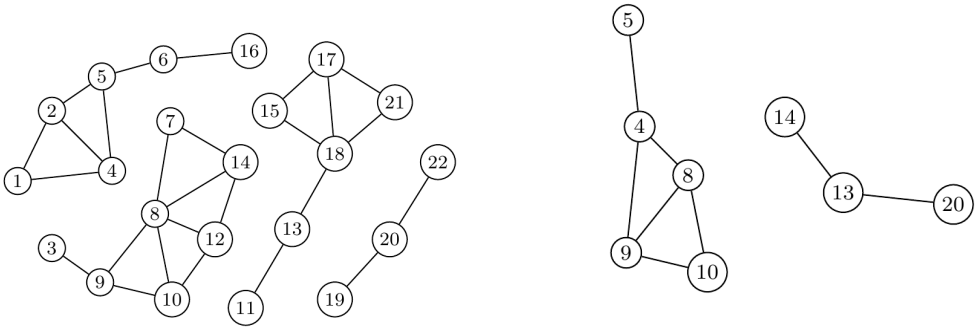
- $v_1,\ v_2\ \in V \setminus \bigcup_{j=1}^{K'} y'_j$;
- $v_1 \backsim v_2 \Leftrightarrow \exists\ j \in \{1, \ldots, K\}$ such that there is a path between v_1 and v_2 in $y_j \setminus \bigcup_{j=1}^{K'} y'_j$.

Example

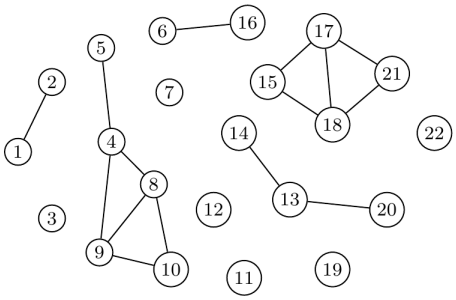
A graph $G = (V, E)$ with $n = 22$ and $m = 49$,



$Y = \{\{1, 2, 4, 5, 6, 16\}, \{3, 7, 8, 9, 10, 12, 14\}, \{11, 13, 15, 17, 18, 21\}, \{19, 20, 22\}\}$ $Y' = \{\{4, 5, 8, 9, 10\}, \{13, 14, 20\}\}$



The new partition



Input

The first line of the input contains two integers n and m (separated by a single space) where n is the number of vertices and m the number of edges of the graph. The following m lines contain each two integers i and j that specify the edge that connects the node i to the node j in the graph.

The remaining lines define the partition Y and the set of non overlapping subgraph Y'.

The first line contains an integer K that specify the number of connected subgraphs that define Y. The following K lines are composed by a sequence of integer, separated by a single space, and ended by the integer -1. Each sequence defines the nodes that form one connected subgraph.

The next line specifies the number K' that defines the number of subgraphs considered in Y'. The last K' lines are composed of a sequence of integers, separated by a single space and ended by the integer -1. Each sequence defines the nodes that form one subgraph of Y'.

Output

The new partition, using the following format.

First, one line containing an integer K'', followed by K'' lines, each one containing an ordered sequence of integers (separated by a single space, from the smallest integer to the biggest) that form a connected subgraph of the solution.

Theses sequences are lexicographically sorted.

Constraints

- $2 \leq n \leq 100$;
- $2 \leq m \leq 200$;

- $2 \leq K \leq 50$;
- $2 \leq K' \leq 50$;

Sample Input

```
22 49
1 2
1 4
1 3
2 5
2 4
3 4
3 9
4 5
4 7
4 8
4 9
5 6
5 7
6 16
6 7
7 15
7 14
7 16
7 8
8 14
8 12
8 10
8 9
9 10
10 12
10 11
11 13
11 12
11 19
12 14
12 13
13 19
13 18
13 14
13 20
14 15
14 18
15 16
15 17
15 18
16 17
17 18
17 21
```

```
18 20
18 21
18 22
19 20
20 22
21 22
4
1 2 4 5 6 16 -1
3 7 8 9 10 12 14 -1
11 13 15 17 18 21 -1
19 20 22 -1
2
4 5 8 9 10 -1
13 14 20 -1
```

Sample Output

```
11
1 2
3
4 5 8 9 10
6 16
7
11
12
13 14 20
15 17 18 21
19
22
```

Problem F: Heirs and the cost of inheritances

A very rich man owned several fields that over the years became very unprotected and damaged through neglect of their surrounding fences. One day, the local authorities decided to force him to pay for the maintenance of the fences around these fields. Seeing the bill, and despite his great wealth – after all, for him, wealth was a matter of saving – the man had a heart attack and left the bill to his heirs.

In order to keep owning the fields, the heirs were obliged to pay the bill. And so the dispute between them began. After much strife they decided to contact you to help them to devise a fair way of sharing the costs.

Task

For each field (i.e. for each test case) you have to compute a fair share of the cost of the repair of the fence.

One field is defined here by n points (that form the field). The fence around the field can be seen as a sequence $\{l_1, \ldots, l_n\}$ (of length n) of line segments. Each line segment l_i has a (positive integer) distance d_i. Hence, the fence is given to you as a ordered sequence of n integer distances (from the first d_1 to the last d_n). The heirs decide that each heir should be responsible for the upkeep of one or more of these segments.

Let p be the number of heirs. A share $\{c_1 \ldots c_p\}$ is a p-partition of the n line segments that form the field such that each partition contains only *consecutive* line segments.

For the sake of simplicity we only consider here partitions in which the first component starts with the first line segment l_1 (and, thus, the last component ends with the last line segment l_n). For instance

$$\underbrace{c_1 = \{l_1, l_2, l_3\}, c_2 = \{l_4, l_5\}, c_3 = \{l_6, l_7, l_8, l_9\} \ldots c_p = \{l_{n-2}, l_{n-1}, l_n\}}_{p \text{ components}}$$

is a possible share.

The length L_c of a component $c = \{l_i, \ldots, l_j\}$ of a share is $d_i + d_{i+1} + \ldots + d_j$.

An optimal solution – i.e. an optimal share – is a solution that minimizes the length of the longest component (and so, the more expansive). Because there can be more than one optimal share, your task is only to output the length of the longest component of an optimal share.

Input

The input is composed by $n + 2$ lines. Each line contains exactly one positive integer.
The first line contains p, the number of heirs.
The second line contains n, the number of line segments.
The subsequent n lines contain the distances d_i, one per line, from d_1 to d_n.

Output

One line with the (integer) length of the longest component of the computed optimal share.

Constraints

Be aware, the man has many many heirs and was very rich ... the fields may be very large, with many many segments.

- The maximum number of heirs p to consider is 500.
- The maximum number of line segments n to consider is 1000.
- $n > p$

- for $i \in \{1..n\}, 0 < d_i \leq 100$.

... You have to be much more clever than the heirs.

Input example

```
5
7
4
3
4
3
3
5
3
```

Output example

```
7
```