PROBLEM 1: Gene Assembly

File Names

Source File: assembly.c, assembly.pas, etc.
Input File: assembly.in
Output File: assembly.out

Statement of the Problem

With the large amount of genomic DNA sequence data being made available, it is becoming more important to find genes (parts of the genomic DNA which are responsible for the synthesis of proteins) in these sequences. It is known that for eukaryotes (in contrast to prokaryotes) the process is more complicated, because of the presence of *junk DNA* that interrupts the coding regions of genes in the genomic sequence. That is, a gene is composed by several pieces (called *exons*) of coding regions. It is known that the order of the exons is maintained in the protein synthesis process, but the number of exons and their lengths can be arbitrary.

Most gene finding algorithms have two steps: in the first they search for possible exons; in the second they try to assemble a largest possible gene, by finding a chain with the largest possible number of exons. This chain must obey the order in which the exons appear in the genomic sequence. We say that exon i *appears* before exon j if the end of i precedes the beginning of j.

The objective of this problem is, given a set of possible exons, to find the chain with the largest possible number of exons that could be assembled to generate a gene.

Input Format

Several input instances are given. Each instance begins with the number $0 < n < 1000$ of possible exons in the sequence. Then, each of the next n lines contains a pair of integer numbers that represent the position in which the exon starts and ends in the genomic sequence. You can suppose that the genomic sequence has at most 50000 basis. The input ends with a line with a single 0.

Output Format

For each input instance your program should print in one line the chain with the largest possible number of exons, by enumerating the exons in the chain. If there is more than one chain with the same number of exons, your program can print anyone of them.

Sample Input

```
6
340 500
220 470
100 300
880 943
525 556
612 776
3
705 773
124 337
453 665
0
```

Sample Output

```
3 1 5 6 4
2 3 1
```

PROBLEM 2: Genetic Combinations

File Names

Source File: combin.c, combin.pas, etc.
Input File: combin.in
Output File: combin.out

Statement of the Problem

The population of the small village of Zombiniville is famous for its genetic characteristics. The inhabitants present only 5 variations on 4 aspects (hair, eyes, nose and feet). Two research companies, Gentec and Genco, tabulated an extensive series of data on the inhabitants of Zombiniville.

Both companies used symbols for the 4 aspects and its variations. Gentec used the groups (A,B,C,D,E), (F,G,H,I,J), (K,L,M,N,O) and (P,Q,R,S,T), while Genco used (a,b,c,d,e), (f,g,h,i,j), (k,l,m,n,o) and (p,q,r,s,t), but not necessarily for the same groups or in the same order inside a group. For example, the group (A,B,C,D,E) from Gentec could correspond to (r,p,t,s,q) from Genco. In other words, "A" corresponds to "r", "B" corresponds to "p", "C" to "t", "D" to "s" and "E" to "q".

Recently the two companies reached an agreement and decided to exchange their research results. However, a group of international activists against human genetic research invaded the offices of the two companies and destroyed part of the reports - exactly the interpretation of the codes.

You and your team were hired to develop a program to establish a relationship among the two notations. As the companies don't want to waste its capital at random, they prepared a battery of tests for your program, before buying it.

Input Format

Each test case has the following data:

1. A line with n, the number of individuals in the test case. You can assume that $0 < n < 64$.
2. A line with n descriptions (one for each individual) in the Gentec notation. Each notation is constituted by 4 characters, the first in the interval A-E, the second in the interval F-J, the third in K-O and the last in P-T. A notation is separated from the following by exactly one blank space.
3. A line with n descriptions (for the same individuals, in the same order) in the Genco notation. Each notation is also constituted by 4 characters, the first in the interval a-e, the second in the interval f-j, the third in k-o and the last in p-t. Again, each notation is separated from the following by exactly one blank space.

After a test case, a new one begins. The end of test cases is indicated by $n = 0$.
You can assume that the test cases are well formed. There are no errors in the input data.

Output Format

For each test case, an association table should be supplied, as in the example below. Note that the pairs are printed in alphabetical order of the capital letters, five per line. Obviously, Gentec and Genco are testing your program, and they don't supply their real data. You can assume that the association table is changed between test cases. You should print only the associations that are unequivocally determined. If there are two or more possibilities for a given letter, you should print a question mark after this letter.

Sample Input

```
5
AGNP AFNQ BHMP AFNP AGKQ
egmr dgnr ehlp egnr dgms
12
AGNP AFNP BHMP AFNT EFKR CHKQ AGNP AILR DHKT BHKT EIKP BGOR
agnp afnp bhmp afnt efkr chkq agnp ailr dhkt bhkt eikp bgor
0
```

Sample Output

```
Test #1:
A-g B-h C-? D-? E-?
F-n G-m H-l I-? J-?
K-s L-? M-n N-r O-?
P-e Q-d R-? S-? T-?
Test #2:
A-a B-b C-c D-d E-e
F-f G-g H-h I-i J-j
K-k L-l M-m N-n O-o
P-p Q-q R-r S-s T-t
```

PROBLEM 3: Palindrom Numbers

File Names

Source File: pal.c, pal.pas, etc.
Input File: pal.in
Output File: pal.out

Statement of the Problem

We say that a number is a **palindrom** if it is the same when read from left to right or from right to left. For example, the number 75457 is a palindrom.

Of course, the property depends on the basis in which the number is represented. The number 17 is not a palindrom in base 10, but its representation in base 2 (10001) is a palindrom.

The objective of this problem is to verify if a set of given numbers are palindroms in any basis from 2 to 16.

Input Format

Several integer numbers comprise the input. Each number $0 < n < 50000$ is given in decimal basis in a separate line. The input ends with a zero.

Output Format

Your program must print the message `Number i is palindrom in basis` where `i` is the given number, followed by the basis where the representation of the number is a palindrom. If the number is not a palindrom in any basis between 2 and 16, your program must print the message `Number i is not palindrom`.

Sample Input

```
17
19
0
```

Sample Output

```
Number 17 is palindrom in basis 2 4 16
Number 19 is not palindrom
```

PROBLEM 4: Robotic Jigsaw

File Names

Source File: jigsaw.c, jigsaw.pas, etc.
Input File: jigsaw.in
Output File: jigsaw.out

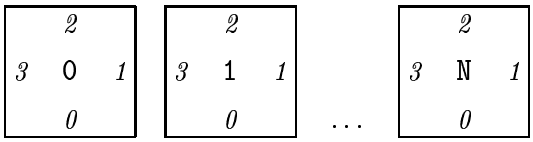
Statement of the Problem

Your task here is to write a program to aid in the assembly of a jigsaw puzzle by a robot.

The puzzle is a rectangular array of flat four-sided pieces. Except for the outer borders, each side of each piece is shaped in such a way that it matches one side of exactly one other piece.

When the puzzle is presented to the robot, all the pieces are scrambled and spread out on a table, face up, in front of the robot's camera eye. Let's assume that someone already wrote software that does the image processing part of the problem: find the puzzle pieces in the image captured by the camera, break the outline of each piece into its four sides, and find pairs of sides that have the same shape. Your task is to program the next step — namely, decide where in the grid each piece should be placed, and how it should be rotated.

The image-processing software assigns, to each piece that the robot sees on the table, a unique *piece number* between 0 and $N-1$, in some arbitrary order. It also labels the sides of the piece with integers 0, 1, 2, 3, in counterclockwise order, starting from an arbitrary side:

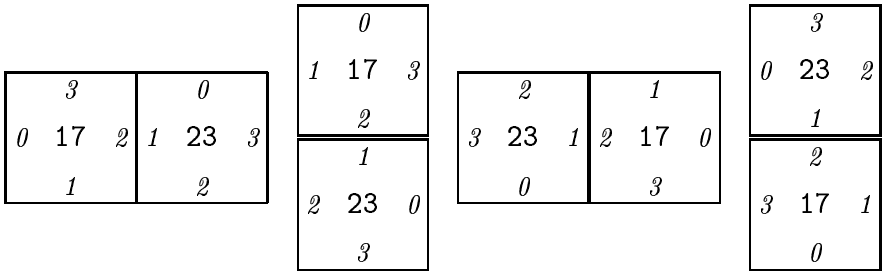


Input Format

The input file contains several consecutive instances of the problem. Each instance consists of: one line contaning two integers $N, 0 < N < 500$ (the number of pieces) and K; then come K lines, one for each pair of similar sides that were found among all pieces seen by the robot, in arbitrary order. Each of these lines contains four integers like this:

```
17 2    23 1
```

This line says that side number 2 of piece number 17 fits side 1 of piece 23. Therefore, in the assembled puzzle, pieces 17 and 23 should be neighbors, in one of these four positions:



You can assume that the input data includes enough information to assemble the puzzle completely. Note, however, that it may not include all pairs of matching sides, and it may include some redundant pairs that could be deduced from the previous ones. The end of the input is marked by a line containing a single 0.

Output Format

For the i^{th} instance of the input file you should write the line `Instance i:`, followed by a sequence of N lines, one for each piece, containing four integers such as `6 9 17 2`. This line says that piece number 17 should be placed in row 6 and column 9 of the grid, turned so that its side number 2 lies at the bottom. (By convention, rows are numbered from top to bottom, and columns from left to right, starting with 0.)

Note also that any solution can be turned by multiples of 90 degrees to give three other valid solutions. In order to make the output unique, you must turn your solution around so that piece number 0 has its side number 0 at the bottom (turned towards the robot). In any case, the upper left corner of the assembled puzzle should be at position $[0, 0]$.

The lines of the output should be presented in lexicographic order of the rows and columns, as shown in the sample output.

Sample Input

```
12   13
0 0    5 0
0 2    6 2
1 2   11 0
1 3    9 2
2 0    5 1
2 1    4 2
3 0   10 3
3 3    8 0
4 0    7 1
4 1   11 1
6 1    7 2
8 2    9 3
10 0   11 2
0
```

Sample Output

```
Instance 1:
    0    0     3 3
    0    1    10 0
    0    2     7 1
    0    3     6 2
    1    0     8 2
    1    1    11 0
    1    2     4 2
    1    3     0 0
    2    0     9 1
    2    1     1 0
    2    2     2 3
    2    3     5 2
```

This represents the solution below.

	1			*2*			*3*			*0*	
2	3	*0*	*3*	10	*1*	*0*	7	*2*	*1*	6	*3*
	3			*0*			*1*			*2*	
	0			*2*			*0*			*2*	
1	8	*3*	*3*	11	*1*	*1*	4	*3*	*3*	0	*1*
	2			*0*			*2*			*0*	
	3			*2*			*1*			*0*	
0	9	*2*	*3*	1	*1*	*2*	2	*0*	*1*	5	*3*
	1			*0*			*3*			*2*	

PROBLEM 5: Direct Subtraction

File Names

Source File: subtract.c, subtract.pas, etc.
Input File: subtract.in
Output File: subtract.out

Statement of the Problem

Some algorithms on image processing are more efficient when applied to small patterns, such as 3×3 matrices. One way of decomposing a given figure into small components is to apply the operation of direct subtraction, which is described in the following.

Given a 0/1 matrix $A_{m \times m}$ and a 0/1 matrix $B_{3 \times 3}$ we define the matrix $C_{(m-2)\times(m-2)} = A \ominus B$ obtained in the following way:

$$c_{ij} = \begin{cases} 1 & \text{if matrix } B \text{ can be subtracted from the } 3 \times 3 \text{ submatrix} \\ & \text{of } A \text{ centered in } (i+1, j+1) \text{ without generating negative numbers} \\ 0 & \text{otherwise} \end{cases}$$

We call matrix C a **valid direct subtraction** of B from A if for every $a_{ij} = 1$ in matrix A, there is a 1 in matrix C which results from a subtraction of B from a submatrix of A containing a_{ij}, and the element of B which is subtracted from a_{ij} is equal to 1.

Example: Given matrices A and B,

$$A = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

the direct subtraction $C = A \ominus B$ is **valid** and is given by

$$C = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Now, given matrices A and B,

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} B = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

the direct subtraction $C = A \ominus B$ is **not valid** and is given by

$$C = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

The objective of this problem is to determine if a matrix A can be transformed into a 3×3 matrix through a sequence of valid direct subtractions of, possibly different, 3×3 matrices.

Input Format

Several input instances are given. Each instance begins with the dimension $0 < n < 20$ of the matrix to be decomposed. The following n lines describe the rows of that matrix, as a sequence of n 0's and 1's, with no blank spaces between them. The input ends with a line with a single 0.

Output Format

For each input instance your program must identify it by printing `Instance i` (where `i` is the number of the instance) and, in the next line a message `Yes` or `No` for the case, resp. that the matrix is (resp. is not) decomposable.

Sample Input

```
5
01010
11111
01111
00111
00010
5
10001
00000
00100
00000
10001
0
```

Sample Output

```
Instance 1
Yes
Instance 2
No
```

PROBLEM 6: Points Within

File Names

Source File: within.c, within.pas, etc.
Input File: within.in
Output File: within.out

Statement of the Problem

Several drawing applications allow us to draw polygons and almost all of them allow us to fill them with some color. The task of filling a polygon reduces to knowing which points are inside it, so programmers have to colour only those points.

You're expected to write a program which tells us if a given point lies inside a given polygon described by the coordinates of its vertices. You can assume that if a point is in the border of the polygon, then it is in fact inside the polygon.

Input Format

The input file may contain several instances of the problem. Each instance consists of: (i) one line containing integers N, $0 < N < 100$ and M, respectively the number of vertices of the polygon and the number of points to be tested. (ii) N lines, each containing a pair of integers describing the coordinates of the polygon's vertices; (iii) M lines, each containing a pair of integer coordinates of the points which will be tested for "withinness" in the polygon.

You may assume that: the vertices are all distinct; consecutive vertices in the input file are adjacent in the polygon; the last vertex is adjacent to the first one; and the resulting polygon is simple, that is, every vertex is incident with exactly two edges and two edges only intersect at their common endpoint. The last instance is followed by a line with a 0 (zero).

Output Format

For the i^{th} instance in the input file, you have to write one line in the output file with the phrase "Problem i:", followed by several lines, one for each point tested, in the order they appear in the input file. Each of these lines should read "Within" or "Outside", depending on the outcome of the test. The output of two consecutive instances should be separated by a blank line.

Sample Input

```
3 1
0 0
0 5
5 0
10 2
3 2
4 4
3 1
1 2
1 3
2 2
0
```

Sample Output

```
Problem 1:
Outside

Problem 2:
Outside
Within
```