

Rujia Liu's Present

Contest 2

A Big Contest of Brute Force

Airplane Scheduling

Be Together Again and Forever

Chinese Mahjong

Digital Logic

Editing a Book

Flipull

Guarding the Chessboard

How Many Numbers?

Irrigation

Jewel Magnetizer

KTV

June 2nd, 2007

Problem A

Airplane Scheduling

Most airports have a big flat land for airplanes to land and take off. Here is an example, where the flat land is divided into 4 rows and 5 columns. Gray squares are places to land and take off, black squares are obstacles, and squares with numbers are parking spaces.

01				02
03	04		05	06
07	08			09
10	11			12

Each airplane is assigned to a parking space. It can choose an arbitrary gray square to land on, then goes to its assigned parking space by a sequence of horizontal and vertical movements (each movement is moving one square north, south, east or west). It cannot move diagonally, nor can it move into an obstacle square or a parking space that is occupied by another airplane. Empty squares without numbers are always free to move on. After reaching its assigned parking space, the airplane waits until it's time for it to take off. Then, it goes to an arbitrary gray square to take off (not necessarily the one it landed on). Initially the flat land is empty. An assignment is feasible if every landing and taking off can be accomplished. Note that different planes might be assigned to the same parking space, as long as the schedule is feasible.

The event list is represented by a sequence of integers, where positive means landing, negative means taking off. It is guaranteed that each plane lands and takes off exactly once (at the end, the flat land is empty again).

For example, the event list $+1, +2, +3, +4, +5, +6, -6, -5, -4, -3, -2, -1$ has a feasible assignment $12, 09, 05, 06, 02, 10$, which are the parking spaces assigned to airplane 1, 2, 3, 4, 5, 6, respectively.

Write a program to assign parking spaces to airplanes.

Input

The input consists of at most 20 test cases. Each case begins with a line containing n, r and c ($0 < n < 21, 2 < r, c < 11$), where n is the number of airplanes, r and c are the number of rows and columns of the flat land. The next r lines each contains c pairs of characters separated by a single space. Each pair is either a landing space(==), an empty space(. .), an obstacle square(##) or a parking space(two digits). Different parking spaces have different numbers. The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and whether there is a solution. If there is, the second line should contain n two-digit integers (with leading zeros if any), the assigned parking space number of the corresponding airplane. If more than one solution exists, any one is acceptable.

Rujia Liu's Present 2: A Big Contest of Brute Force

Sample Input

```
6 4 5
01 .. == .. 02
03 04 .. 05 06
07 08 .. ## 09
10 11 .. .. 12
+1 +2 +3 +4 +5 +6 -6 -5 -4 -3 -2 -1
3 3 3
== .. 01
.. ## 02
.. ## 03
+1 +2 +3 -3 -1 -2
0
```

Output for the Sample Input

```
Case 1: Yes
12 09 05 06 02 10

Case 2: No
```

Source: *Chinese National Olympiad in Informatics 1996*
Adapted by Rujia Liu, with test data enhanced

Problem B

Be Together Again and Forever

Jiajia and WinD quarreled. They got angry with each other. They went far from each other. They regretted. They missed each other so much. They wanna say sorry. They wanna meet in a city as soon as possible. They wanna be together again and forever.

Each of their routes should be simple - no city should be reached more than once. Their routes should be disjoint - no city should be reached by them both, except the last one - their meeting city. They don't want anyone to travel more, so the distance covered by them should be exactly the same.

Input

The input consists of at most 20 test cases. Each case begins with a line containing four integers n, m, J, W ($0 < n < 201$, $0 < m < 3n/2$, $1 \leq J, W \leq n$), the number of cities and bidirectional roads in the city network, and the initial city number of Jiajia and WinD. Each of the following m lines describes a road. There are three integers in each road description: i, j, d , indicating there is a bidirectional road connected i and j , with length d ($1 \leq i, j \leq n$, $1 \leq d \leq 10$). Each pair of cities can be connected by at most one road. It is assumed that every pair of cities can be reached from each other. J does not equal to W . The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the distance covered by each person in the first line. The next two lines should contain the route plan for Jiajia and WinD, respectively. Each route plan begins with an integer c , the number of cities visited. The next c integers form the list of cities visited. There is always a solution.

Sample Input

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

Output for the Sample Input

```
Case 1: 2
3 1 2 3
2 4 3

Case 2: 5
2 1 3
3 4 2 3
```

Source: Balkan Olympiad in Informatics, 1998
Adapted by Rujia Liu, with test data enhanced

Problem C

Chinese Mahjong

Mahjong (麻将) is a game of Chinese origin usually played by four persons with tiles resembling dominoes and bearing various designs, which are drawn and discarded until one player wins with a hand of four combinations of three tiles each and a pair of matching tiles.

A set of Mahjong tiles will usually differ from place to place. It usually has at least 136 tiles, most commonly 144, although sets originating from America or Japan will have more. The 136-tile Mahjong includes:

Dots: named as each tile consists of a number of circles. Each circle is said to represent copper (tong) coins with a square hole in the middle. In this problem, they're represented by 1T, 2T, 3T, 4T, 5T, 6T, 7T, 8T and 9T.



Bams: named as each tile (except the 1 Bamboo) consists of a number of bamboo sticks. Each stick is said to represent a string (suo) that holds a hundred coins. In this problem, they're represented by 1S, 2S, 3S, 4S, 5S, 6S, 7S, 8S and 9S.



Craks: named as each tile represents ten thousand (wan) coins, or one hundred strings of one hundred coins. In this problem, they're represented by 1W, 2W, 3W, 4W, 5W, 6W, 7W, 8W and 9W.



Wind tiles: East, South, West, and North. In this problem, they're represented by DONG, NAN, XI, BEI.



Dragon tiles: red, green, and white. The term dragon tile is a western convention introduced by Joseph Park Babcock in his 1920 book introducing Mahjong to America. Originally, these tiles are said to have something to do with the Chinese Imperial Examination. The red tile means you pass the examination and thus will be appointed a government official. The green tile means, consequently you will become financially well off. The white tile (a clean board) means since you are now doing well you should act like a good, incorrupt official. In this problem, they're represented by ZHONG, FA, BAI.



There are $9 \times 3 + 4 + 3 = 34$ kinds, with exactly 4 tiles of each kind, so there are 136 tiles in total.



To who may be interested, the 144-tile Mahjong also includes:



Flower tiles: typically optional components to a set of mahjong tiles, often contain artwork on their tiles. There are exactly one tile of each kind, so $136 + 8 = 144$ tiles in total. In this problem, we don't consider these tiles.



Rujia Liu's Present 2: A Big Contest of Brute Force

Chinese Mahjong is very complicated. However, we only need to know very few of the rules in order to solve this problem. A **meld** is a certain set of tiles in one's hand. There are three kinds of melds you need to know (to who knows Mahjong already, **kong** is not considered):

Pong: A set of three identical tiles. Example: ; .

Chow: A set of three suited tiles in sequence. All three tiles must be of the same suites. Sequences of higher length are not permissible (unless it forms more than one meld). Obviously, wind tiles and dragon tiles can never be involved in chows. Example: ; .

Eye: The pair, while not a meld, is the final component to the standard hand. It consists of any two identical tiles.

A player wins the round by creating a standard mahjong hand. That means, the hand consists of an eye and several (possible zero) pongs and chows. Note that each tile can be involved in exactly one eye/pong/chow.

When a hand is one tile short of winning, the hand is said to be a ready hand, or more figuratively, 'on the pot'. The player holding a ready hand is said to be waiting for certain tiles. For example



is waiting for ,  and .

To who knows more about Mahjong: don't consider special winning hands such as 七对子.

Input

The input consists of at most 50 test cases. Each case consists of 13 tiles in a single line. The hand is legal (e.g. no invalid tiles, exactly 13 tiles). The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and a list of waiting tiles sorted in the order appeared in the problem description (1T~9T, 1S~9S, 1W~9W, DONG, NAN, XI, BEI, ZHONG, FA, BAI). Each waiting tile should be appeared exactly once. If the hand is not ready, print a message 'Not ready' without quotes.

Sample Input

```
1S 1S 2S 2S 2S 3S 3S 3S 7S 8S 9S FA FA
1S 2S 3S 4S 5S 6S 7S 8S 9S 1T 3T 5T 7T
0
```

Output for the Sample Input

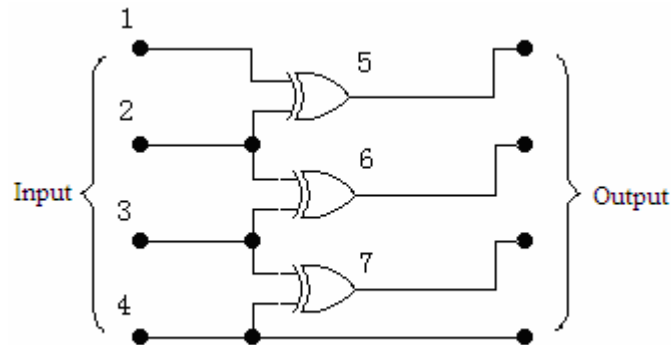
```
Case 1: 1S 4S FA
Case 2: Not ready
```

Problem D

Digital Logic

Your task is to design a 4-input 4-output digital circuit, given some 2-input 1-output gates.

Every gate is described by three 0-1 integers Y_{00} , Y_{01} and Y_{11} , the output of the gate when there are exactly 0, 1 and 2 of the inputs are set to 1. Note that all the gates are symmetric, so if there is exactly one input is set to 1, the output is the same no matter which one is set. Make sure that both inputs of each gate are connected to the output of another gate or a signal source, since the output will be unpredictable if at least one input is dangling. Be aware that the output of a gate can never go back to its input either directly or indirectly (i.e. the circuit should not contain a directed cycle).



To make your design as simple as possible, you should use minimal number of gates. It is guaranteed that the circuit could be designed with at most 6 gates.

Input

The input consists of at most 30 test cases. Each case contains a single integer n ($n < 6$), indicating there are n kinds of gates. Each of the n lines contained four integers, m_i , Y_{00} , Y_{01} , Y_{11} , the number of available gates of this kind, the output values when exactly 0, 1, 2 inputs are set to 1. There are at most 10 gates in total (i.e. the sum of m_i will not exceed 10). The next line contains 16 integers Y_{0000} , Y_{0001} , Y_{0010} , ..., Y_{1111} , the output values for each possible input combination. Y_{pqrs} 's binary form is the output of the circuit when the four inputs are p , q , r , s respectively. That is, if Y_{pqrs} 's binary form is $abcd$, then the four outputs are a , b , c , d respectively. The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and an integer p , the minimal number of gates required. The next p lines each contains four integers s , k , a and b , where s is the serial number of the gate (inputs of the entire circuit are numbered 1~4, gates are numbered 5~ $p+4$), k is the type number of the gate (gate types are numbered 1~ n in the same order as they appeared in the input), a and b are serial numbers of two inputs of the gate. It should be satisfied that $a < s$ and $b < s$. The last line should contain four integers: the serial numbers of the gates of four outputs (should be all between 1 and $p+4$). There should be exactly one empty line after each case.

Sample Input

```
1
5 0 1 0
0 3 6 5 12 15 10 9 8 11 14 13 4 7 2 1
1
2 1 1 0
8 12 10 14 9 13 11 15 8 12 10 14 1 5 3 7
0
```

Output for the Sample Input

```
Case 1: 3
5 1 1 2
6 1 2 3
7 1 3 4
5 6 7 4

Case 2: 1
5 1 2 1
5 4 3 2
```

Problem E

Editing a Book

You have n equal-length paragraphs numbered 1 to n . Now you want to arrange them in the order of 1, 2, ..., n . With the help of a clipboard, you can easily do this: Ctrl-X (cut) and Ctrl-V (paste) several times. You cannot cut twice before pasting, but you can cut several contiguous paragraphs at the same time - they'll be pasted in order.

For example, in order to make {2, 4, 1, 5, 3, 6}, you can cut 1 and paste before 2, then cut 3 and paste before 4. As another example, one copy and paste is enough for {3, 4, 5, 1, 2}. There are two ways to do so: cut {3, 4, 5} and paste after {1, 2}, or cut {1, 2} and paste before {3, 4, 5}.

Input

The input consists of at most 20 test cases. Each case begins with a line containing a single integer n ($1 < n < 10$), the number of paragraphs. The next line contains a permutation of 1, 2, 3, ..., n . The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the minimal number of cut/paste operations.

Sample Input

```
6
2 4 1 5 3 6
5
3 4 5 1 2
0
```

Output for the Sample Input

```
Case 1: 2
Case 2: 1
```

Source: ACM/ICPC Kanpur Site 2001-2002

Adapted by Rujia Liu

Problem F

Flipull

You're an orange blob on a ladder on the right of the screen. You fire blocks to the left, which eliminates blocks of the same kind, then flips back the first block of a different kind. The goal is to have at most b blocks left in the game. The game continues until you don't have legal moves. If you still have more than b blocks when the game ends, you lost.

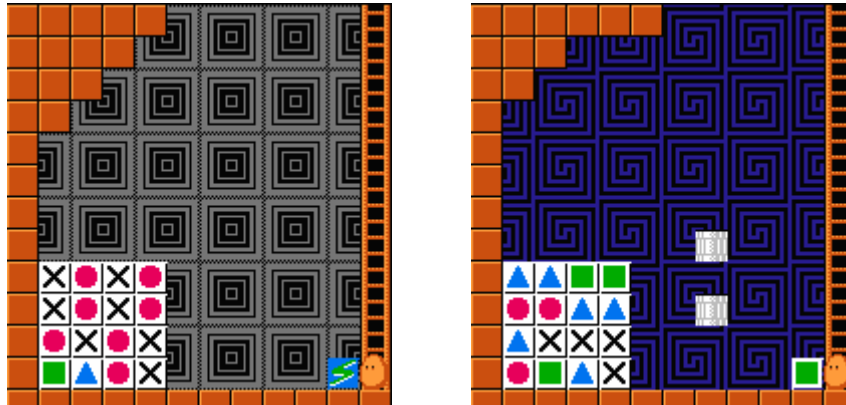


Fig 1. Game of Flipull

There are 4 kinds of blocks: the blue triangle (T), the pink circle (O), the green square (#) and the cross (X). After firing a block, it keeps flying to the left until it meets a block of a different kind, or reaches a wall. In the former case, the new block is flipped back; in the latter case, the block is reflected and moves down. When moving down, the block flips back the first block of different kind as usual. If the block reaches the bottom wall, it returns to you. Note that each fire must eliminate at least one block, so you can't flip back a block directly by firing at it. As you may guess, a block falls down immediately when the block below it is eliminated, with the help of the gravity. There is also a special magic block which may only appear initially on your hand. It will change into the first block it fires to, so it could be regarded as a sort of 'wildcard' block, as shown in figure 1(a).

The initial blocks always form a square of $4*4$, $5*5$ or $6*6$, on the bottom-left corner of the game. The rows are numbered $r1$ to $r6$ from bottom to top, and the columns are numbered $c1$ to $c6$ from left to right. There are 12 steps in the ladder (you may convince yourself by counting in figure 1), numbered 1 to 12 from the bottom (the current positions in figure 1) to the top. Since walls can reflect blocks, firing at the 12 steps are actually moving along $r1, r2, r3, r4, c1, c1, c1, c1, c2, c3, c4, X$ in figure 1(a), where X means firing at step 12 will not touch any block (thus an illegal move). Figure 1(b) is a little bit different: pipes can also reflect blocks, so firing at the 12 steps are actually moving along $r1, r2, X, r4, X, c1, c1, c1, c1, c2, c3, X$. Note that $r3$ and $c4$ can never be accessed.

Here is a complete example of how the game illustrated in figure 1(a) is solved (by leaving exactly 3 blocks). Fire four blocks at step 8, 9, 10, 11 (figure 2(a)), then fire at step 10 (figure 2(b)) and step 2 (figure 2(c)), finally step 1.

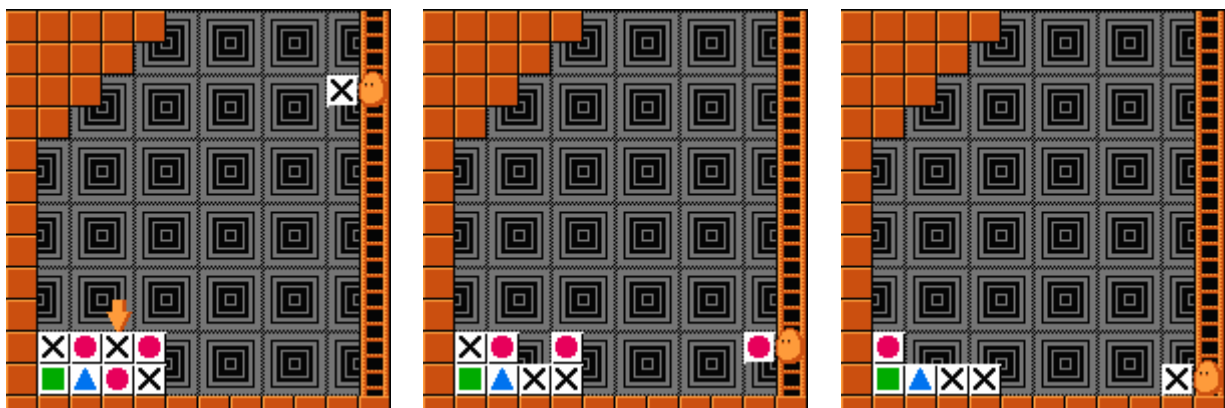


Fig 2. Solving figure 1(a)

Write a program to solve the game of Flipull with minimal number of fires.

Rujia Liu's Present 2: A Big Contest of Brute Force

Input

The input consists of at most 50 test cases. Each case begins with a line containing n , b and c ($3 < n < 7$, $2 < b < n^2$), where n is the dimension of the initial blocks, b is the maximal number of blocks left, c is the initial block you're holding (M indicates the magic block). The second line contains the name of the game, which contains at most 100 characters and will not contain white characters (spaces and TABs). The next n lines describe the initial blocks (there will never be a magic block inside). The last line contains the target rows/columns for each step. The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the name of the game in the first line. The second line should contain a single integer k , the minimal number of fires needed in the first line. The third line should contain k integers from 1 to 12, the step numbers for each fire. Print a blank line after each case. If more than one solution exists, any one is acceptable. It is guaranteed that every puzzle is solvable in no more than 20 moves.

Sample Input

```
4 3 M
First
XOXO
XOXO
OXOX
#TOX
r1 r2 r3 r4 c1 c1 c1 c1 c2 c3 c4 X
4 3 #
Pipes
TT##
OOTT
TXXX
O#TX
r1 r2 X r4 X c1 c1 c1 c1 c2 c3 X
4 3 T
NoGreedy!!!
X##T
TXXT
TTT#
OXOT
r1 r2 r3 r4 c1 c1 c1 c1 c2 c3 c4 X
4 3 O
Tough...
XOTO
O##O
TXOO
X#TO
r1 r2 r3 r4 c1 c1 c1 c1 c2 c2 c4 X
0
```

Output for the Sample Input

```
Case 1: First
7
8 9 10 11 10 2 1

Case 2: Pipes
8
4 11 1 2 1 4 2 2

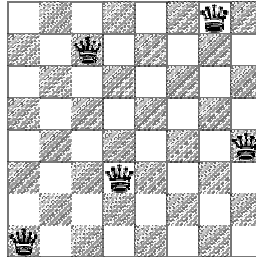
Case 3: NoGreedy!!!
7
11 4 3 11 2 1 2

Case 4: Tough...
10
1 4 1 3 11 1 4 1 3 2
```

Problem G

Guarding the Chessboard

Given an $n*m$ chessboard with some marked squares, your task is to place as few queens as possible to guard (attack or occupy) all marked squares. Below is a solution to an $8*8$ board with every square marked. Note that queens can be placed on non-marked squares.



Input

The input consists of at most 15 test cases. Each case begins with a line containing two integers n, m ($1 < n, m < 10$), the size of the chessboard. Next n lines each contain m characters, 'X' denotes marked square, '.' denotes unmarked squares. The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the minimal number of queens needed.

Sample Input

```
8 8
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX
XXXXXXXXX
8 8
X.....
.X.....
..X.....
...X....
....X...
.....X..
.....X.
.....X
0
```

Output for the Sample Input

```
Case 1: 5
Case 2: 1
```

Problem H

How Many Numbers?

You might have heard the game of 24: given 4 integers, you're to make an expression to get the number 24. For example, given 4, 4, 10, 10, you can write $(10*10-4)/4=24$, given 1, 5, 5, 5, you can write $(5-1/5)*5=24$.

In this problem, your task is a little bit harder: count the number of numbers that can be made. Don't forget to count negative numbers and non-integers. You can use binary additions, subtractions, multiplications and divisions with parenthesis (unary operations are not allowed). Numbers cannot be concatenated to form a larger number (e.g. you cannot concatenate 1 and 2 to get 12).

For example, given two 1's, exactly 3 numbers can be made: $1+1=2$, $1-1=0$, $1*1=1$. You cannot get 11 or -1.

Input

The input consists of at most 30 test cases. Each case begins with a line containing a single integer n ($1 < n < 7$), the number of integers given. The next line contains n non-negative integers not greater than 10. The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the number of numbers that can be made.

Sample Input

```
2
1 1
3
1 4 7
4
1 2 3 5
0
```

Output for the Sample Input

```
Case 1: 3
Case 2: 47
Case 3: 255
```

Problem I

Irrigation

You have a piece of land, which is divided into an $n*m$ grid. In some squares there is a water source capable of providing water for a certain number of other squares. A water source can provide water for zero or more consecutive squares directly to the north, zero or more consecutive squares directly to the east and so on.

Water is precious, so you have just enough water to irrigate the whole piece of land. So it's crucial to design an irrigation system so that every square is either a water source or irrigated by exactly one water source.

Here is an example. Water sources are represented in the form **id(water)**. So the 4-th water is capable of provide water for 5 other squares.

	3(4)			
				6(2)
1(2)		4(5)		
				5(2)
	2(4)			

Fig 1. An irrigation system

Input

The input consists of at most 50 test cases. Each case begins with a line containing three non-negative integers n , m and k ($1 \leq n, m \leq 30$, $1 \leq k \leq 200$), where k is the number of water sources. In the following k lines water sources are described, by three non-negative integers x , y , c ($1 \leq x \leq n$, $1 \leq y \leq m$), the coordinate and water capacity (west-south corner is $(1,1)$), one line for each water source. It is guaranteed that $nm - k = c_1 + c_2 + \dots + c_k$ (just enough water). The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number in the first line, then k lines followed, one for each water source. Each water source is described by four non-negative integers n , e , s , w , the amount of squares irrigated in each direction. It is guaranteed that at least one solution exists. Print a blank line after each test case.

Rujia Liu's Present 2: A Big Contest of Brute Force

Sample Input

```
2 2 2
1 1 1
2 2 1
5 5 6
1 3 2
2 1 4
2 5 4
3 3 5
5 2 2
5 4 2
0
```

Output for the Sample Input

```
Case 1:
1 0 0 0
0 0 1 0

Case 2:
1 0 1 0
1 2 0 1
0 2 1 1
1 2 1 1
0 0 1 1
1 0 0 1
```

Source: *Winter Camp in Yugoslavia 2001.*
Adapted by Rujia Liu

Problem J

Jewel Magnetizer

The block world is divided into n rows n columns of blocks. Rows are numbered 1 to n from top to bottom, and columns are numbered 1 to n from left to right. You start from a square, follow a strictly height-decreasing path until no legal moves are possible. During your trip, you can use a jewel magnetizer to get jewels not too far away. To be precise, you can get a jewel in square (r_1, c_1) if and only if there is a square (r_2, c_2) on your path such that $\max\{|r_1-r_2|, |c_1-c_2|\} \leq r$.

There is one thing you should be aware of: your bag capacity is limited, so you can only get at most m jewels. There is at most one jewel on each block.

1	1	1	1	1
1	6	9	5	1
1	3	10	3	1
1	2	1	2	1
1	1	1	1	1

(a) height

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

(b) jewel value

The picture above shows an example. The numbers in the left picture describes the heights in each block, where the numbers in the right picture describes the jewel values in each block. An optimal solution for $m=5$ and $r=1$ is shown in the figures. The arrows denote the path, gray squares corresponds to jewels you should get. The total value is $4+3+2+2+1=12$.

Write a program to get the highest total value of jewels.

Input

The input consists of at most 30 test cases. Each case begins with a line containing three non-negative integers n , m and r ($1 < n < 21$, $0 < m < 101$, $r < 6$), where n is the size of the block world, m is the bag capacity, r is the range of jewel magnetizer. The next n lines each contain n non-negative integers not greater than 8000, the heights of each block. The next n lines each contain n non-negative integers not greater than 1000, the value of jewel at each block (zero means there is no jewel). The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the highest value.

Rujia Liu's Present 2: A Big Contest of Brute Force

Sample Input

```
5 5 1
3 3
1 1 1 1 1
1 6 9 5 1
1 3 10 3 1
1 2 1 2 1
1 1 1 1 1
4 1 1 1 2
1 1 1 1 1
1 2 3 1 2
1 2 1 1 1
1 1 1 1 1
4 2 1
1 3
4 5 6 9
3 9 9 9
2 1 1 9
9 9 9 9
0 0 0 1
0 0 0 0
0 0 0 0
0 0 1 5
0
```

Output for the Sample Input

```
Case 1: 12
Case 2: 2
```

*Source: Hunan Team Selection Contest for Chinese National Olympiad in Informatics 2002.
Adapted by Rujia Liu, with test data enhanced*

Problem K

KTV

One song is extremely popular recently, so you and your friends decided to sing it in KTV. The song has 3 characters, so exactly 3 people should sing together each time (yes, there are 3 microphones in the room). There are exactly 9 people, so you decided that each person sings exactly once. In other words, all the people are divided into 3 disjoint groups, so that every person is in exactly one group.



However, some people don't want to sing with some other people, and some combinations perform worse than others combinations. Given a score for every possible combination of 3 people, what is the largest possible score for all the 3 groups?

Input

The input consists of at most 1000 test cases. Each case begins with a line containing a single integer n ($0 < n < 81$), the number of possible combinations. The next n lines each contains 4 positive integers a, b, c, s ($1 \leq a < b < c \leq 9, 0 < s < 10000$), that means a score of s is given to the combination (a,b,c) . The last case is followed by a single zero, which should not be processed.

Output

For each test case, print the case number and the largest score. If it is impossible, print -1.

Sample Input

```
3
1 2 3 1
4 5 6 2
7 8 9 3
4
1 2 3 1
1 4 5 2
1 6 7 3
1 8 9 4
0
```

Output for the Sample Input

```
Case 1: 6
Case 2: -1
```