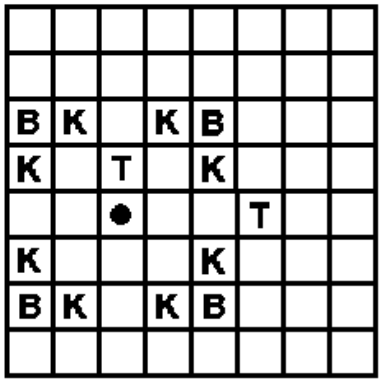
633 A Chess Knight

Presumably everybody knows how a knight can move on a chessboard. One may agree that its movements are quite monotonous, so to make them more entertaining let's define a so called "dynamic knight". A dynamic knight can perform many different movements that may belong to three types:

- type K: two fields forward (in any direction) and one sidewise - like "regular knight";
- type B: two fields diagonally - more like a bishop;
- type T: sort of teleportation to a field which is a mirror reflection with respect to any of two axes of symmetry of the chessboard (we take into consideration only axes of symmetry parallel to sides of the chessboard);

The picture below shows all possible movements of a knight divided into three types K, B and T. Obviously our knight, like the "regular" one cannot move outside the chessboard.



For a dynamic knight it is not relevant whether the fields between the starting field and ending one are occupied or not (again like for the "regular knight"). It only matters whether the ending field is empty. Then the movement can be performed. There has to be a restriction among so many capabilities of a dynamic knight. It cannot perform the same sort of movements consecutively (just not to fall into routine).

Having redefined a chess knight, why not to redefine a chessboard? Our chessboard will be a square of size $2N \times 2N$. N can be any integer number from the range of 3..20. There can be several obstacles of any shape on a chessboard so a knight cannot stop on these defected fields.

Your task is to write a program which can calculate the minimal number of movements to get the knight from one given field to another one. It may be assumed that the first movement can be of any type.

Input

First line contains the number N, being the size of a chessboard. The second one contains field coordinates separated by space character, which is a knight current standpoint. Upper left-hand size corner has coordinates (1, 1). Third line contains destination field for the knight. Consecutive lines contain obstacle coordinates and the line with coordinates (0, 0) ends the obstacle description. Input can contain several sets of data.

Input's end is shown as a line defining chessboard's size as 0.

3171 - Oreon

Asia - Manila - 2006/2007

In the 25th century, civilization is struck by a series of calamities that eventually led mankind to build walled cities interconnected by tunnel bridges to facilitate transportation. Each walled city possesses a unique ore required to build and repair all infrastructure including the tunnels. This material which when combined with other ores from all others cities form an almost indestructible material called ``oreon".

Outside the walled cities are uncivilized barbarians armed with antiquated but destructive weaponry which can effectively shoot down any air transport, but only damage and not penetrate tunnel bridges. Thus each city is interconnected to more than one city in order to have access redundancy in case one of its interconnecting tunnels is damaged.

If a tunnel is damaged, it becomes impassable and would require a substantial amount of ``oreon" to repair the damage. When a single city is made isolated, meaning all of its interconnections are damaged, ``oreon" cannot be manufactured which may lead to the eventual destruction of the wall fortifying the city. You, being the head of the homeland defense unit, are tasked to ensure that all cities remain accessible even by at least a single interconnecting tunnel at all times. Faced with only a limited manpower in the defense unit, you have to determine which tunnel to protect using the least number of people and ensure that no city will be isolated.

Figure 2 shows a map of the walled cities, their interconnecting tunnels and the number of security personnel.

=6in

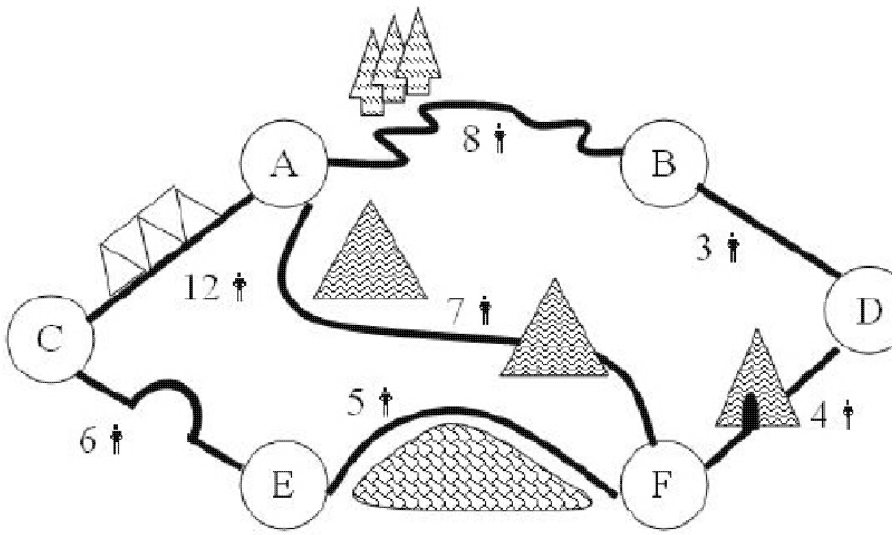


Figure 2: Map of six cities and its interconnecting tunnels

Input

The input will contain several test cases. The first line will indicate the number of test cases. Each test case begins with a number representing the number of walled cities. Cities are labeled alphabetically using the letters in the English alphabet. The subsequent lines contain the number of security personnel needed to protect the tunnel connecting each city to all other cities. A value of zero implies no security personnel needed since no tunnel exists. You are to output which tunnel should be protected and how many personnel are needed for each tunnel.

Output

The output shows the tunnel connection which is named after the cities that it connects (in alphabetical order) and the number of personel needed to protect the tunnel. Order the records in increasing order of presonal. In case two tunnels have the same number of personel, write them in lexicographycal order.

Sample Input

```
1
6
0, 8, 12, 0, 0, 7
8, 0, 0, 3, 0, 0
12, 0, 0, 0, 6, 0
0, 3, 0, 0, 0, 4
0, 0, 6, 0, 0, 5
7, 0, 0, 4, 5, 0
```

Sample Output

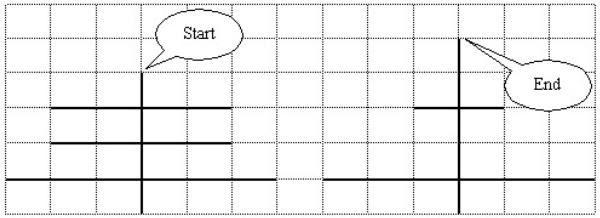
```
Case 1:
B-D 3
D-F 4
E-F 5
C-E 6
A-F 7
```

Manila 2006-2007

Problem B
Big Big Trees

Input: standard input
Output: standard output
Time Limit: 2 seconds
Memory Limit: 16 MB

There are **n** trees arranged in a straight line in the forest, the adjacent two trees are **m** meters apart from each other. The **i**-th tree is $\mathbf{h_i}$ meters high, and at every integer height $\mathbf{x(1<=x<=h_i)}$, the tree has two big leaves with a length of $\mathbf{l_{i,j}}$ on both sides. The two leaves are left-right symmetrical, so each tree is left-right symmetrical too. The picture below shows two trees. Note that the longest leaf should be shorter than **m/2** meters. So every two leaves from different trees can NEVER overlap in left-to-right direction.



A monkey from the top of the 1st tree wants to reach to top of the last tree. He may climb up and down, and he may walk from left to right on the leaves or on the ground. He may also jump from the RIGHT ENDPOINT of a leaf to the LEFT ENDPOINT of another leaf on the next tree(Warning: the monkey cannot jump from or land inside a leaf or the ground! What's more, he jumps along a straight line, so the line should not contain any points of any other leaves). But he may do that only if the distance between the two endpoints are not longer than **k** meters. Clearly, the monkey should always use exactly **n-1** jumps, but he may wall on different route in order to minimize the total distance he walks(NOT climbs). Help him to calculate the minimal total distance walked.

Input

The first line of the input is a single integer **t(1<=t<=10)**, indicating the number of test cases. Each case begins with a line containing three integers **n,m,k(1<=n,m,k<=1000,**), indicating the number of trees, the distance between two adjacent trees, and the maximal allowed jump distance. There are **n** lines following. Each line describes a tree. The first integer **h(1<=h<=20)** is the height of the tree. There are **h** integers following: $\mathbf{l_1, l_2, ... l_h}$ **(0<=l_i<m/2, i=1,2...n)**,

indicating the lengths of the leaves.

Output

For each case, print a number **d** in a single line indicating the minimal total distance the monkey walks.

Sample Input

```
2
2 7 3
4 3 2 2 0
5 3 0 1 0 0
3 50 40
4 15 3 16 10
8 12 12 12 21 12 15 6 14
13 15 23 20 18 14 1 21 9 9 18 23 10 4
```

Sample Output

```
5
28
```

The 2nd OIBH Online Programming Contest. Author: Rujia Liu