- Problems: There are 8 problems (29 pages in all) in this packet.

- Problem Input: Input to each problem are done through the input file. Input filenames are given in the Problem Information Sheet.

- Problem Output: All output should be directed to standard output (screen output).

- Time Limit: The judges will run each submitted program with certain time limit (given in the Problem Information Sheet).

Table 1: Problem Information Sheet

	Problem Name	Input File	Time Limit
Problem A	Tiling Up Blocks	pa.in	30 secs.
Problem B	Quad Trees	pb.in	30 secs.
Problem C	The Suspects	pc.in	30 secs.
Problem D	The Geodetic Set Problem	pd.in	30 secs.
Problem E	Cave Raider	pe.in	30 secs.
Problem F	Gap Punishment Alignment Problem	pf.in	30 secs.
Problem G	Space AI Bombs	pg.in	30 secs.
Problem H	Merging Sequences Problem	ph.in	30 secs.

Problem A
Tiling Up Blocks
Input File: pa.in

Michael The Kid receives an interesting game set from his grandparent as his birthday gift. Inside the game set box, there are n tiling blocks and each block has a form as follows:
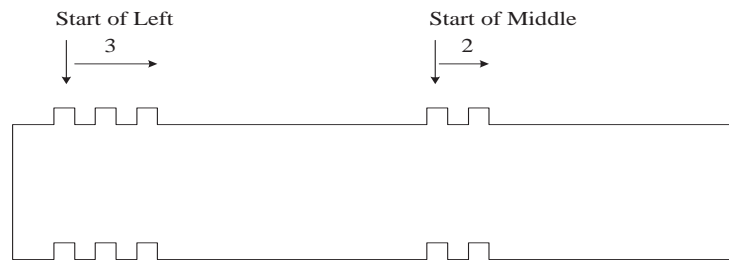

Figure 1: Michael's Tiling Block with parameters (3,2).

Each tiling block is associated with two parameters (ℓ, m), meaning that the upper face of the block is packed with ℓ protruding knobs on the left and m protruding knobs on the middle. Correspondingly, the bottom face of an (ℓ, m)-block is carved with ℓ caving dens on the left and m dens on the middle.

It is easily seen that an (ℓ, m)-block can be tiled upon another (ℓ, m)-block. However, this is not the only way for us to tile up the blocks. Actually, an (ℓ, m)-block can be tiled upon another (ℓ', m')-block if and only if $\ell \geq \ell'$ and $m \geq m'$.

Now the puzzle that Michael wants to solve is to decide what is the tallest tiling blocks he can make out of the given n blocks within his game box. In other words, you are given a collection of n blocks $B = \{b_1, b_2, \ldots, b_n\}$ and each block b_i is associated with two parameters (ℓ_i, m_i). The objective of the problem is to decide the number of tallest tiling blocks made from B.

Input Format: Several sets of tiling blocks. The inputs are just a list of integers. For each set of tiling blocks, the first integer n represents the number of blocks within the game box. Following n, there will be n lines specifying parameters of blocks in

B; each line contains exactly two integers, representing left and middle parameters of the i-th block, namely, ℓ_i and m_i. In other words, a game box is just a collection of n blocks $B = \{b_1, b_2, \ldots, b_n\}$ and each block b_i is associated with two parameters (ℓ_i, m_i).

Note that n can be as large as 10000 and ℓ_i and m_i are in the range from 1 to 100. An integer $n = 0$ (zero) signifies the end of input.

Output Format: For each set of tiling blocks B, output the number of the tallest tiling blocks can be made out of B. Output a single star '*' to signify the end of outputs.

Sample Input

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

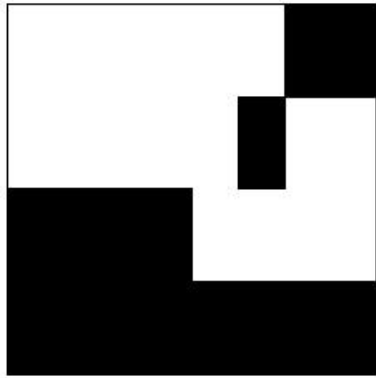## Output for the Sample Input

```
2
3
*
```

Problem B
Quad Trees
Input File: pb.in

A binary image, such as the one shown in Figure 2(a), is usually represented as an array of binary entries, i.e., each entry of the array has value 0 or 1. Figure 2(b) shows the array that represents the binary image in Figure 2(a). To store the binary image of Figure 2(b), the so-called *quad tree partition* is usually used. For an $N \times N$ array, $N \leq 512$ and $N = 2^i$ for some positive integer i, if the entries do not have the same value, then it is partitioned into four $N/2 \times N/2$ arrays, as shown in Figure 2(c). If an $N/2 \times N/2$ array does not have the same binary value, such as the upper right and lower right $N/2 \times N/2$ arrays in Figure 2(c), then we can divide it into four $N/4 \times N/4$ arrays again. These $N/4 \times N/4$ arrays in turn can also, if needed, be divided into four $N/8 \times N/8$ arrays, etc.. The quad tree partition is completed when the whole array is partitioned into arrays of various size in which each array contains only one binary value. Figure 2(c) contains the arrays after the quad tree partition is completed.

Instead of storing the binary image of Figure 2(a), we only need to store the quad tree in the form as Figure 2(d) which is encoded from Figure 2(c). In Figure 2(d), each node represents an array of Figure 2(c) in which the root node represents the original array. If the value of a node in the tree is 1, then it means that its corresponding array needs to be decomposed into four smaller arrays. Otherwise, a node will have a pair of values and the first one is 0. It means that its corresponding array is not necessary to decompose any more. In this case, the second value is 0 (respectively, 1) to indicate that all the entries in the array are 0 (respectively, 1). Thus, we only need to store the tree of Figure 2(d) to replace storing the binary image of Figure 2(a). The way to store the tree of Figure 2(d) can be represented by the following
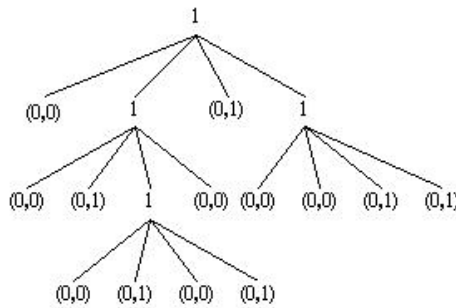
0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

(a)   (b)

0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	1	0	0
0	0	0	0	0	1	0	0
1	1	1	1	0	0	0	0
1	1	1	1	0	0	0	0
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

(c)   (d)

Figure 2: A binary image (a), its array representation (b), its quad tree partition (c), and its quad tree representation (d).

code: (1)(0,0)(1)(0,1)(1)(0,0)(0,1)(1)(0,0)(0,0)(0,0)(0,1)(0,1)(0,0)(0,1)(0,0)(0,1). This code is just to list the values of the nodes from the root to leaves and from left to right in each level. Deleting the parentheses and commas, we can obtain a binary number 100101100011000000010100010001 which is equal to 258C0511 in hexadecimal. You are asked to design a program for finding the resulting hexadecimal value for each given image.

Input Format: There is an integer number k, $1 \le k \le 100$, in the first line to

indicate the number of test cases. In each test case, the first line is also a positive integer N indicating that the binary image is an $N \times N$ array, where $N \leq 512$ and $N = 2^i$ for some positive integer i. Then, an $N \times N$ binary array is followed in which at least one blank is between any two elements.

Output Format: The bit stream (in hexadecimal) used to code each input array.

Sample Input

```
3
2
0 0
0 0
4
0 0 1 1
0 0 1 1
1 1 0 0
1 1 0 0
8
0 0 0 0 0 0 1 1
0 0 0 0 0 0 1 1
0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0
1 1 1 1 0 0 0 0
1 1 1 1 0 0 0 0
1 1 1 1 1 1 1 1
1 1 1 1 1 1 1 1
```

Output for the Sample Input

```
0
114
258C0511
```

Problem C
The Suspects
Input File: pc.in

Severe acute respiratory syndrome (SARS), an atypical pneumonia of unknown aetiology, was recognized as a global threat in mid-March 2003. To minimize transmission to others, the best strategy is to separate the suspects from others.

In the Not-Spreading-Your-Sickness University (NSYSU), there are many student groups. Students in the same group intercommunicate with each other frequently, and a student may join several groups. To prevent the possible transmissions of SARS, the NSYSU collects the member lists of all student groups, and makes the following rule in their standard operation procedure (SOP).

> Once a member in a group is a suspect, all members in the group are suspects.

However, they find that it is not easy to identify all the suspects when a student is recognized as a suspect. Your job is to write a program which finds all the suspects.

Input Format: The input file contains several cases. Each test case begins with two integers n and m in a line, where n is the number of students, and m is the number of groups. You may assume that $0 < n \le 30000$ and $0 \le m \le 500$. Every student is numbered by a unique integer between 0 and $n-1$, and initially student 0 is recognized as a suspect in all the cases. This line is followed by m member lists of the groups, one line per group. Each line begins with an integer k by itself representing the number of members in the group. Following the number of members, there are k integers representing the students in this group. All the integers in a line are separated by at least one space.

A case with $n = 0$ and $m = 0$ indicates the end of the input, and need not be processed.

Output Format: For each case, output the number of suspects in one line.

Sample Input

```
100 4
2 1 2
5 10 13 11 12 14
2 0 1
2 99 2
200 2
1 5
5 1 2 3 4 5
1 0
0 0
```

Output for the Sample Input

```
4
1
1
```

Problem D
The Geodetic Set Problem
Input File: pd.in

Let $G = (V, E)$ be a connected graph without loops and multiple edges, where V and E are the vertex and edge, respectively, sets of G. For any two vertices $u, v \in V$, the *distance* between vertices u and v in G is the number of edges in a shortest u-v path. A shortest path between u and v is called a u-v *geodesic*. Let $I(u, v)$ denote the set of vertices such that a vertex is in $I(u, v)$ if and only if it is in some u-v geodesic of G and, for a set $S \subseteq V$, $I(S) = \bigcup_{u,v \in S} I(u, v)$. A vertex set D in graph G is called a *geodetic set* if $I(D) = V$. The *geodetic set problem* is to verify whether D is a geodetic set or not. We use Figure 3 as an example. In Figure 3, $I(2, 5) = \{2, 3, 4, 5\}$ since there are two shortest paths between vertices 2 and 5. We can see that vertices 3 and 4 are lying on one of these two shortest paths respectively. However, $I(2, 5)$ is not a geodetic set since $I(2, 5) \neq V$. Vertex set $\{1, 2, 3, 4, 5\}$ is intuitively a geodetic set of G. Vertex set $D = \{1, 2, 5\}$ is also a geodetic set of G since vertex 3 (respectively, vertex 4) is in the shortest path between vertices 1 and 5 (respectively, vertices 2 and 5). Thus, $I(D) = V$. Besides, vertex sets $\{1, 3, 4\}$ and $\{1, 4, 5\}$ are also geodetic sets. However, $D = \{3, 4, 5\}$ is not a geodetic set since vertex 1 is not in $I(D)$.

Input Format: The input file consists of a given graph and several test cases. The first line contains an integer n indicating the number of vertices in the given graph, where $2 \leq n \leq 40$. The vertices of a graph are labeled from 1 to n. Each vertex has a distinct label. The following n lines represent the adjacent vertices of vertex i, $i = 1, 2, \cdots, n$. For example, the second line of the sample input indicates that vertex 1 is adjacent with vertices 2 and 3. Note that any two integers in each line are separated by at least one space. After these n lines, there is a line which contains the
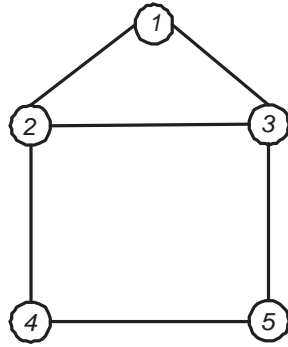
Figure 3: A graph G.

number of test cases. Each test case is shown in one line and represents a given subset D of vertices. You have to determine whether D is a geodetic set or not.

Output Format: For each test case, output 'yes' in one line if it is a geodetic set or 'no' otherwise.

Sample Input

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

Output for the Sample Input

```
yes
yes
no
yes
yes
no
```

Problem E
Cave Raider
Input File: pe.in

Afkiyia is a big mountain. Inside the mountain, there are many caves. These caves are connected by tunnels. Hidden in one of the caves is a terrorist leader. Each tunnel connects two caves. There could be more than one tunnels connect the same two caves. At the joint of a tunnel and a cave, there is a door. From time to time, the terrorists close a tunnel by shutting the two doors at the two ends, and "clean" the tunnel. It is still a mystery how they clean the tunnel. However, we know that if a person (or any living creature) is trapped in the tunnel when it is being cleaned, then the person (or the living creature) will die. After a cleaning of the tunnel is finished, the door will open, and the tunnel can be used again.

Now the intelligence servicemen have found out which cave the leader is hiding, and moreover, they know the schedule of the cleaning of the tunnels. Jing Raider is going to go into the cave and catch the leader. You need to help him find a route so that he can get to that cave in the shortest time. Be careful not to be trapped in a tunnel when it is being cleaned.

Input Format: The input consists of a number of test cases. The 1st line of a test case contains four positive integers n, m, s, t, separated by at least one space, where n is the number of caves (numbered $1, 2, \cdots, n$), m is the number of tunnels (numbered $1, 2, \cdots, m$), s is the cave where Jing is located at time 0, and t is the cave where the terrorist leader is hiding. ($1 \leq s, t \leq n \leq 50$ and $m \leq 500$).

The next m lines are information of the m tunnels: Each line is a sequence of at most 35 integers separated by at least one space. The first two integers are the caves

that are the ends of the corresponding tunnel. The third integer is the time needed to travel from one end of the tunnel to the other. This is followed by an increasing sequence of positive integers (each integer is at most 10000) which are alternately the closing and the opening times of the tunnel. For example, if the line is

10 14 5 6 7 8 9

then it means that the tunnel connects cave 10 and cave 14, it takes 5 units of time to go from one end to the other. The tunnel is closed at time 6, opened at time 7, then closed again at time 8, opened again at time 9. Note that the tunnel is being cleaned from time 6 to time 7, and then cleaned again from time 8 to time 9. After time 9, it remains open forever.

If the line is

10 9 15 8 18 23

then it means that the tunnel connects cave 10 and cave 9, it takes 15 units of time to go from one end to the other. The tunnel is closed at time 8, opened at time 18, then closed again at time 23. After time 23, it remains closed forever.

The next test case starts after the last line of the previous case. A 0 signals the end of the input.

Output Format: The output contains one line for each test case. Each line contains either an integer, which is the time needed for Jing to get to cave t, or the symbol *, which means that Jing can never get to cave t. Note that the starting time is 0. So if $s = t$, i.e., Jing is at the same cave as the terrorist leader, then the output is 0.

Sample Input

```
2 2 1 2
1 2 5 4 10 14 20 24 30
1 2 6 2 10 22 30
6 9 1 6
1 2 6 5 10
1 3 7 8 20 30 40
2 4 8 5 13 21 30
3 5 10 16 25 34 45
2 5 9 22 32 40 50
3 4 15 2 8 24 34
4 6 10 32 45 56 65
5 6 3 2 5 10 15
2 3 5 2 9 19 25
2 2 1 2
1 2 7 6 9 12
1 2 9 8 12 19
0
```

Output for the Sample Input

```
16
55
*
```

Problem F

Gap Punishment Alignment Problem

Input File: pf.in

Consider two strings $X = x_1 x_2 \cdots x_m$ and $Y = y_1 y_2 \cdots y_n$ over an alphabet set $\Sigma = \{A, G, C, T\}$. Denote $\Sigma^\star = \Sigma \cup \{-\}$, where "$-$" (dash) is the symbol that represents a space (or blank) in strings. A *string alignment* is to align X and Y and form two strings $X^\star, Y^\star$ over the alphabet $\Sigma^\star$ such that:

1. the two strings $X^\star, Y^\star$ have the same lengths, and

2. ignoring dashes, the string $X^\star$ is the same as the string X, and the string $Y^\star$ is the same as the string Y.

As an example, an alignment of two strings 'GATCCGA' and 'GAAAGCAGA' is as follows:

```
G-A--TCCGA
GAAAG-CAGA.
```

There are three gaps in the above alignment; here a *gap* is defined as a string of consecutive dashes. Now, let us consider the following alignment:

```
GA---TCCGA
GAAAG-CAGA.
```

Here are two gaps within this alignment. The rule of measuring the *intermittent gap punishment alignment score* (abbreviated by GPS) is as follows:

19

- If x_i is aligned with y_j, the score $\sigma(x_i, y_j)$ is

$$\sigma(x_i, y_j) = \begin{cases} 2 & \text{if } x_i = y_j \\ -1 & \text{if } x_i \neq y_j \end{cases}$$

- If a (consecutive) subsequence of x_i's (or y_j's) is aligned with a gap of length k, the score is defined as $-(4 + k)$.

That is, in the first alignment example given above, its GPS is $2 - (4 + 1) + 2 - (4 + 2) - (4 + 1) + 2 - 1 + 2 + 2 = -7$. For the second alignment, its GPS is $2 + 2 - (4 + 3) - (4 + 1) + 2 - 1 + 2 + 2 = -3$.

Given two strings, the problem we would like to solve is to find an alignment such that its GPS is maximized. Thus, in our example, the best alignment is

```
GA--TCCGA
GAAAGCAGA.
```

Its GPS is 2+2-(4+2)-1+2-1+2+2=2.

In our problem, m and n are at most 500. Furthermore, it is required that no space in one sequence is aligned with a space in another.

Input Format: The input file format is as follows:

1. The first line contains an integer n of sequence pairs; the number n is at most 50.

2. The 2nd line is the sequence X of the first pair.

3. The 3rd line is the other sequence Y of the first pair.

   $\vdots$

2i. The $(2i)$-th line is the sequence X of the i-th pair.

2i+1. The $(2i + 1)$-th line is the other sequence Y of the i-th pair.

$$\vdots$$

2n. The $(2n)$-th line is the sequence X of the n-th pair.

2n+1. The $(2n + 1)$-th line is the other sequence Y of the n-th pair.

Output Format: For each pair of sequences, output the *maximum* GPS in one line.

Sample Input

3

ACGGCTTAGATCCGAGAGTTAGTAGTCCTAAGCTTGCA

AGCTTAGAAAGCAGACACTTGATCCTGACGGCTTGAA

TTGAGTAGTGTTTTAGTCCTACACGACACATCAAATTCGGACAAGGCCTAGCT

TTCAAGTCCTACAATGTGTGTCAAATTCGCTTGGCCGAAAGCC

TTTGGGAACGTGTGTAGACTTCCCCATGCGATGG

AACACACACGGACTTCATGCTGG

Output for the Sample Input

18

20

2

Problem G

Space AI Bombs

Input File: pg.in

The time is year 3000. Human beings have settled on planets in many solar systems and have a star war with an alien species called *Romulans*. The human scientists design a new weapon called *AI bomb* which is capable of space travel across the vast space. Before launching the weapons, humans send probes to collect Romulan's defense parameters. The data shows that *Romulans* have set up shields in the routes to their home worlds. Fortunately, some secret information reveals that the shield can be penetrated using an ion beam with a particular range of frequency. It is possible to pass the shield if an AI bomb emits an ion beam within that frequency. Now, human scientists have plotted an interstellar map between several human planets and Romulan planets. The map is a directed graph like Figure 4. In the figure, human planets are drawn in boxes (denoted as Hx) and Romulan worlds are drawn in triangles (denoted as Rx), where x is an integer number. A shield is drawn as a circle in the figure (denoted as Sx).

Since humans only know where the shields are but do not know the frequency of each shield, they decide to launch a large number of AI bombs. Each bomb is configured to emit an ion beam at a particular frequency at first. Once an AI bomb passes a shield, it will modulate its frequency to a different value by increasing or decreasing a predefined value. For example, in Figure 4, an AI bomb B1 is launched from H3 with initial frequency 150 and an interval $\pm$ 100. So, when B1 penetrates shield S5, it may modulate its frequency to 50, 250, or keep its previous frequency 150. After that, the bomb can choose any routes available in the star map. In the example, the bomb B1 is possible to reach Romulan homeworld R9 by penetrating S5
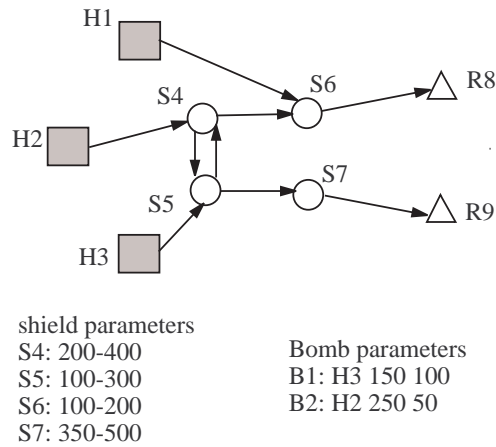
23

Figure 4: An example star map.

with the original frequency 150 and then passing S4 by changing its frequency to 250 and keeping frequency 250 to pass S5 again and by changing its frequency to 350 in order to penetrate S7 and then finally nuking Romulan planet R9.

Unfortunately, Romulans knows what humans are planning. Their spies got the map and the bomb parameters. Of course, Romulans have shield parameters at hand. *They want to know if there are any AI bombs which can reach their homeworlds under current shield settings.* Please note that human AI bombs can choose any route to travel. If an AI bomb has any chance to reach a Romulan's home world, then the bomb must be reported.

Please write a program for the Romulan to defend vicious humans. To simplify the problem, we restrict the frequency values between 0 and 1000. When a bomb's new frequency is outside the range, the new frequency is invalid.

Input Format: The test data begins with a number n in a line which is the number of test cases. In each test case, it begins with two numbers v and e in a line where v is the number of vertices (including human planets, Romulan planets, and shields) and e

is the number of directed edges, $2 \le v \le 100000$ and $2 \le e \le 500000$. For convenience, the vertices are indexed starting with 1.

Next, a line beginning with '`human` m' tells that there are m human planets. Following the string are m integers, which are the indices of human planets.

Same as above, a string '`romulan` k' is used to tell the vertex indices of Romulan's planets.

A string '`shield` x' begins the shield parameters, where x is the number of shields. Each shield parameter is described by $(s\ l\ u)$, where s is the shield's index, l is the lower bound of the range, and u is the upper bound of the range. The values of l and u is between 0 and 1000.

A string '`edge` u' begins the directed edge data, where u is the number of edges. Each edge is described by $(s\ d)$, where s is the index of the source vertex and d is the index of end vertex.

A string '`bomb` p' begins with the data of deployed AI bombs, where p is the number of bombs and $1 \le p \le 10000$. Each bomb is described by $(h\ f\ i)$, where h is the index of a vertex (i.e., a human planets where the bomb is located), f is the initial frequency, and i is the interval to be increased/decreased.

Output Format: Please output the number of bombs that can possibly reach any of the Romulan homeworlds in one line for each test case. Note that, a bomb may be able to reach more than one Romulan planets. In that case, it is still counted as 1.

Sample Input

```
1

9 9

human 3 1 2 3

romulan 2 8 9

shield 4

4 200 400

5 100 300

6 100 200

7 350 500

edge 9

1 6

6 8

2 4

4 6

4 5

5 4

3 5

5 7

7 9

bomb 2

3 150 100

2 250 50
```

Output for the Sample Input

```
2
```

Problem H

Merging Sequences Problem

Input File: ph.in

Let S be a sequence of n integers, where $S[k]$ with $1 \leq k \leq n$ denotes the k-th number of S. The *maximum prefix sum* of S, denoted $h(S)$, is defined to be

$$h(S) = \max_{0 \leq j \leq n} \sum_{1 \leq k \leq j} S[k].$$

(Note that the range for j starting from 0 is to ensure $h(S) \geq 0$, because $\sum_{1 \leq k \leq 0} S[k] = 0$.) For example, if

$$W = -2, 1, -3;$$

$$X = 1, 2, 4, 3, -1, -5, 2, 0, -1, 3, -2;$$

$$Y = -1, 2, 0, 1, 3, -5, 3, 2, 4, -2, -1,$$

then $h(W) = 0$, $h(X) = 1+2+4+3 = 10$ and $h(Y) = -1+2+0+1+3-5+3+2+4 = 9$.

For each $i = 1, 2, \ldots, \ell$, let S_i be a sequence of n_i integers. We say that a sequence S of n numbers is a *merged* sequence of $S_1, S_2, \ldots, S_\ell$ if the following conditions hold.

1. $n = n_1 + n_2 + \cdots + n_\ell$.

2. There is a 1–1 mapping f from $\{1, 2, \ldots, n\}$ to $\{(i, j) \mid 1 \leq i \leq \ell \text{ and } 1 \leq j \leq n_i\}$ such that if $f(t) = (i, j)$ then $S[t] = S_i[j]$.

3. If $t < t'$, $f(t) = (i, j)$ and $f(t') = (i, j')$, then $j < j'$.

For example, if we have

$$S_1 = 1, 3, -5, 2, -2;$$

$$S_2 = 2, 4, -1;$$

$$S_3 = -1, 0, 3,$$

27

then both X and Y above are merged sequences of S_1, S_2, S_3. The following sequence, however, is not a merged sequence of S_1, S_2, S_3.

$$Z = 1, 3, -5, 2, -2, 2, 4, -1, -1, 3, 0.$$

(Clearly, if the last two numbers 3 and 0 in Z are exchanged, then the resulting sequence is a merged sequence of S_1, S_2, S_3.)

Your job is to produce a merged sequence S^* of $S_1, S_2, \ldots, S_\ell$ with minimum $h(S^*)$. For instance, the following sequence is a merged sequence for the above S_1, S_2, S_3 whose maximum prefix sum is minimized:

$$S^* = -1, 1, 0, 3, -5, 2, -2, 2, 4, -1, 3.$$

One can verify that $h(S^*) = -1 + 1 + 0 + 3 - 5 + 2 - 2 + 2 + 4 - 1 + 3 = 6$.

Input Format: The first line contains a number m with $1 \leq m \leq 10$ indicating the number of test cases. Each of the next m lines lists a test case. Each test case lists those ℓ ($1 \leq \ell \leq 5$) input sequences separated by numbers 9999. Each test case ends with a number -9999. Two consecutive numbers in a sequence are separated by at least one single space. You may assume that each input sequence consists of at most 100 integers, each of which is between -100 and 100.

Output Format: For each test case $S_1, S_2, \ldots, S_\ell$, output its $h(S^*)$ in a single line.

Sample Input

```
3
1 3 -5 2 -2 9999 2 4 -1 9999 -1 0 3 -9999
5 1 1 9999 -2 -2 -2 9999 10 -20 -9999
-2 1 -3 -9999
```

Output for the Sample Input

```
6
4
0
```

Problem I
Harmonic Periods
Input File: pi.in

In real-time scheduling, predictability is very important, i.e., we would like to know the whole schedule before we really run the tasks. Rate-monotonic scheduling is very popular in real-time scheduling for periodic tasks, where tasks with shorter periods have higher priority. However, it is still difficult to know the start time and finish time of each task and they might be different in each period, especially for tasks with low priority, i.e. long period. The hyperperiod, the least common multiple of all periods, is usually too big to be practical to describe the whole schedule. However, if the task periods are harmonic, i.e. are multiples, it is possible to find the start time and finish time of each task quickly because the schedule becomes more regular.
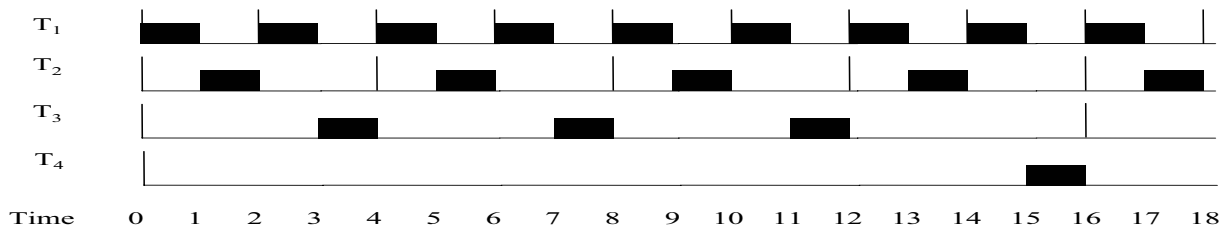

Figure 5: task schedule example

Figure 1 shows that periodic tasks T_1, T_2, T_3, T_4 with execution times 1, 1, 3, 1 and periods 2, 4, 16, 32 respectively are schedulable, each task finishes execution in its period, using the Rate-Monotonic scheduling algorithm since T_1, T_2, T_3, T_4 finish execution at time 1, 2, 12, 16 respectively. T_3 is preempted by T_1 and T_2 at time 4 and time 8 and resume at time 7 and time 11.

Input Format: All the input numbers are positive integers, < 500000, separated by a space or new line. The first line is the number of task sets. Then, the task sets

30

are listed set by set. Each task set is listed by a line of the number of tasks, ≤ 100, and lines of task execution time and period pairs, execution time $<$ period. The periods are harmonic, not sorted, and are different in a task set.

Output Format: For each task set, find and print out the finish time of the task with the largest period using rate-monotonic scheduling, if schedulable; otherwise print out -1.

Sample Input

3

4

1 2

1 4

3 16

1 32

3

1 4

4 8

256 1024

3

1 2

3 8

1 4

Output for the Sample Input

16

1024

-1