The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem A
Block Compaction

We are given *R*, a set of rectangles, which are *x,y*-axis parallel and mutually disjoint, but can share boundaries. All rectangles should be placed in the quadrant $x \geq 0, y \geq 0$ of the plane. We are asked to compact these rectangles using the procedure COMPACT in the following. After this procedure, places of all rectangles are fixed. You are asked to find a smallest enclosing rectangle for them. The enclosing rectangle also must be *x,y*-axis parallel.

Procedure: COMPACT
do {
  Step 1. Move blocks downward until no blocks can be moved.
  Step 2. Move blocks leftward until no blocks can be moved.
} Until no blocks can be moved downward or leftward.

Let us explain this compaction procedure in the following example. Figure 1 is the initial layout of the given set of rectangles. After moving blocks downward as much as possible, we get the layout as shown in Figure 2.

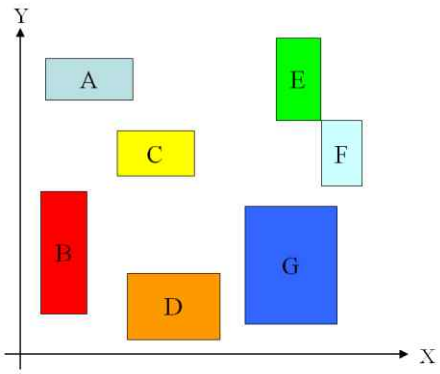


Figure 1. Input Rectangles

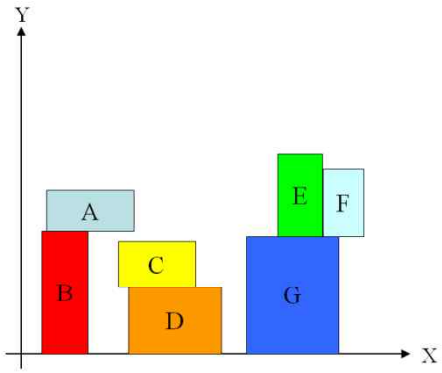


Figure 2. After moving blocks downward

Since we cannot move any blocks downward in Figure 2, we try to move blocks leftward by COMPACT in Figure 3. Note that all blocks must remain mutually disjoint in the COMPACT procedure, but they can be put together along boundary edges. For boundary edges sharing, see {E, G, F}, {A, B} and {C, D} rectangles in Figure 2.

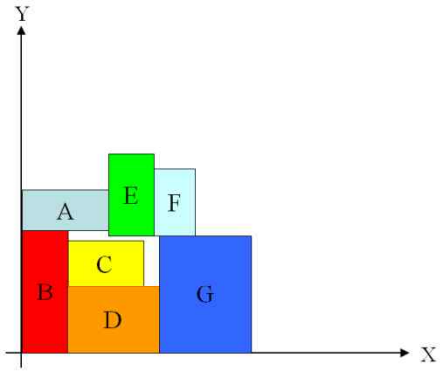


Figure 3. After moving blocks leftward

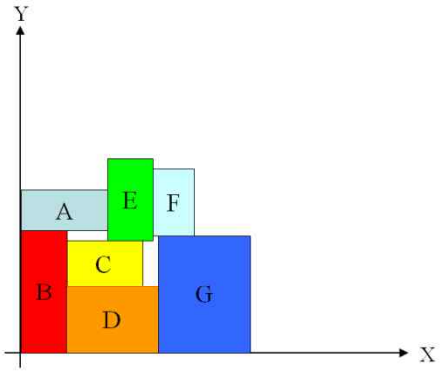


Figure 4. After moving blocks downward

If we repeat this procedure as in Figure 4 until we cannot move any blocks, then we get the final compacted rectangles as shown in Figure 5. Finally we obtain the dotted rectangle as the smallest enclosing rectangle for them.

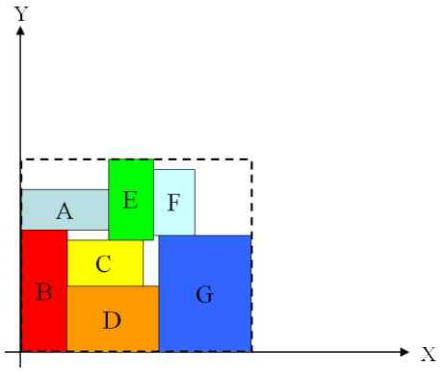


Figure 5. We finally get the smallest enclosing rectangle (dotted box)
by applying COMPACT till no blocks can be moved.

Your task is to compute the final enclosing rectangle obtained from applying COMPACT.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. The first line of each test case contains N ($1 \leq N \leq 500$), the number of rectangles given. Then N lines follow to define each rectangle with the integer coordinates of the lower-left vertex (x,y) and upper-right vertex (p,q) in a single line as `x y p q`, where $x < p$, $y < q$ and $0 \leq x, y, p, q \leq 100{,}000$.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain two numbers W and H, the width and height of the enclosing rectangle obtained by COMPACT.

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 3 10 10 30 30 10 50 15 55 90 10 100 100 4 10001 1001 11001 70001 20001 15001 25001 30001 20001 40001 80001 45001 20000 60000 28500 61000 1 34010 34010 34100 34100	30 90 61000 69000 90 90

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem B
Chemical Products

We are given three chemicals A, B, and C contained in separated bottles. If we mix any two chemicals in equal units, we get a valuable chemical product. We call AB the mixture of A and B, BC the mixture of B and C, and CA the mixture of C and A (if we mix one unit of A and one unit of B, we get one unit of AB). Because the prices of AB, BC, and CA can be different, the profit obtained by making the products can vary depending on the way the products are made. Assume we can mix any two chemicals only in integer units.

For example, suppose there are 5 units of A, 3 units of B, and 7 units of C. The price of each product is shown in the next table.

Chemical Product	Price(per one unit)
AB	$100
BC	$30
CA	$80

If we mix one unit of A and one unit of B, we get $100 by making one unit of AB. Next, mixing 2 units of B and 2 units of C, we get $60. Finally, mixing 4 units of C and 4 units of A, we get $320. The total profit is $480, which is the maximum possible. Note that some portion of one chemical can remain unused: one unit of C is unused in this example.

For another example, suppose there are 3 units of A, 3 units of B, and 3 units of C. Each price of AB, BC, and CA is $100. If we mix 2 units of A and 2 units of B, we get $200. Next, mixing one unit of B and one unit of C, we get $100. Finally, mixing one unit of C and one unit of A, we get $100. The total profit is $400, which is the maximum possible.

Write a program which calculates the maximum profit when the quantities of A, B, C and the prices of AB, BC, CA are given.

Input
Your program is to read from standard input. The input consists of *T* test cases. The number of test cases *T* is given in the first line of the input. Each test case consists of two lines. The first line of each test case contains three integers. The three integers represent the quantities of chemicals A, B, and C. The second line of each test case contains three integers. The three integers represent the prices of chemical products AB, BC, and CA. All integers in the first and second lines are between 1 and 1,000, both inclusive.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain the maximum profit for the test case.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 5 3 7 100 30 80 3 3 3 100 100 100	480 400

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem C
Color Length

Cars painted in different colors are moving in a row on the road as shown in Figure 1. The color of each car is represented by a single character and the distance of two adjacent cars is assumed to be 1. Figure 1 shows an example of such cars on the road. For convenience, the numbers in Figure 1 represent the locations of each car.

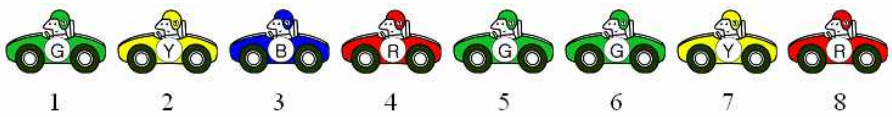


Figure 1. Cars in different colors on the road

For any color c, *location*(c) represents set of the locations of the cars painted in color c and color length L(c) is defined as follows:

$$L(c) = \max location(c) - \min location(c)$$

For example, according to Figure 1, *location*(G) = {1,5,6}, *location*(Y) = {2,7}, *location*(B) = {3}, and *location*(R) = {4, 8}. Hence the color length of each color and the sum of the color lengths are as follows.

Color	G	Y	B	R	Sum
L(c)	5	5	0	4	14

In Gyeongju City, almost all the roads including the main street of the city were constructed more than 500 years ago. The roads are so old that there are a lot of puddles after rain. Visitors have complained about the bad condition of the roads for many years. Due to the limited budget, the mayor of the city decided to repair firstly the main street of the city, which is a four-lane road, two lanes for each direction.

However, since the main street is a backbone of the city, it should not be blocked completely while it is under repair, or it is expected that serious traffic jams will occur on almost all the other roads in the city. To allow cars to use the main street during the repair period, the city decided to block only two lanes, one lane for each direction. Hence, the cars in the two lanes for each direction should merge into a single lane before the blocked zone.

For instance, as shown in Figure 2, cars in the two lanes merge into a single lane as shown in Figure 3. To differentiate the cars in the same color, a unique identifier is assigned to each car.

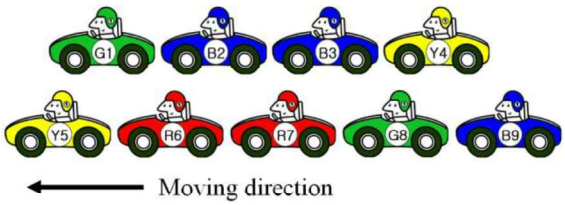


Figure 2. Cars moving in two lanes before merging

Figure 3 shows two different merging scenarios after merging the cars from the two lanes. As shown in Figure 3, cars in the two lanes do not necessarily merge one by one from each lane. The distance between two adjacent cars after merging is also assumed 1.

After merging (Scenario 1):

After merging (Scenario 2):

Figure 3. Two different merging scenarios

For each merging scenario shown in Figure 3, the color length for each color and the sum of the color lengths are as follows:

Color	G	Y	B	R	Sum
L(c): Scenario 1	7	3	7	2	19
L(c): Scenario 2	1	7	3	1	12

As you can imagine, there are many different ways of merging other than the two examples shown in Figure 3.

Given two character strings which represent the color information of the cars in the two lanes before merging, write a program to find the sum of color lengths obtained from the character string, which is the color information of cars after merging, such that the sum of color lengths is minimized.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case consists of two lines. In the first line, a character string of length n ($1 \le n \le 5{,}000$) that is the color information of the cars in one lane before merging is given. In the second line, a character string of length m ($1 \le m \le 5{,}000$) that is the color information of the cars in the other lane is given. Every color is represented as an uppercase letter in English, hence the number of colors is less than or equal to 26.

Output

Your program is to read from standard input. Print exactly one line for each test case. The line should contain the sum of color lengths after merging the cars in the two lanes optimally as described above.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 AAABBCY ABBBCDEEY GBBY YRRGB	10 12

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem D
Equipment

The Korea Defense and Science Institute, shortly KDSI, has been putting constant effort into new equipment for individual soldiers for recent years, and at last released N new types of equipment. KDSI has already done evaluation of each of the N types of equipment, finally resulting in scores in five categories: attack improvement, defense improvement, vision improvement, portability, and easiness of usage. The score in each category is quantified as an integer in the range 0 to 10,000, and the rating of each type of equipment is thus represented as a sequence of five integers.

In consideration of costs and capability of average individual soldiers, KDSI also reported that each soldier will be able to install at most K types of equipment on his body to extend his ability. If a single type is installed on a soldier, then his ability in each category is extended by the specified score of that type. Moreover, if a soldier installs more than one type of equipment, then his ability in each category is extended by the maximum score of the chosen types in that category. For example, if the vision improvement scores of type a and type b are 10 and 15, respectively, then installing a combination of two types a and b will result in a vision improvement by their maximum score 15. We call the maximum score 15 the *extension score of a category*; so, the extension score of vision improvement for combination $\{a, b\}$ is 15 in this example.

KDSI now started devising a way of finding an optimal combination of K types of equipment for best performance of individual soldiers. While a force can sometimes be of a special purpose so that a certain category would be more important than the others, every single category is, however, regarded equally important in general. For this general purpose, KDSI defined the *objective score of a combination* of equipment to be the sum of the extension scores of the five categories for the combination. KDSI thus wants to find a best combination of K types of equipment such that its objective score is maximized among all possible combinations of K types. You are asked by KDSI to devise and write a computer program that finds the objective score of a best combination of K types of equipment, that is, the maximum possible objective score for all possible combinations of K types among the given N types of equipment.

Put differently, you are given N types of equipment $\{1, \ldots, N\}$ and their ratings R_i represented by five integers $R_i = (r_{i,1}, r_{i,2}, r_{i,3}, r_{i,4}, r_{i,5})$ with $0 \le r_{i,j} \le 10{,}000$ for each $i = 1, \ldots, N$ and $j = 1, \ldots, 5$. Given another natural number K ($1 \le K \le N$), your program has to compute the objective score of a best combination of K types of equipment.

For example, consider an input instance in which $N = 4$, $K = 2$, and each R_i is given as below:

$R_1 = (30, 30, 30, 30, 0)$
$R_2 = (50, 0, 0, 0, 0)$
$R_3 = (0, 50, 0, 50, 10)$
$R_4 = (0, 0, 50, 0, 20)$.

Then, choosing R_1 and R_3 forms a best combination of two types $\{1, 3\}$ and yields the objective score 30+50+30+50+10 = 170, which will be the answer of a correct program.

Input

Your program is to read from standard input. The input consists of T test cases. The number T of test cases is given in the first line of the input. From the second line, each test case is given in order, consisting of the following: a test case contains two integers N ($1 \leq N \leq 10{,}000$) and K ($1 \leq K \leq N$) in its first line, and is followed by N lines each of which consists of five integers inclusively between 0 and 10,000, representing the five scores $r_{i,1}$, $r_{i,2}$, $r_{i,3}$, $r_{i,4}$, and $r_{i,5}$ of each type i of equipment for $i = 1, \ldots, N$ in order. Two consecutive integers in one line are separated by a single space and there is no empty line between two consecutive test cases.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain a single integer that is the objective score of a best combination of K types of equipment; the maximum possible objective score for all possible combinations of K types among the given N types of equipment for the corresponding test case.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 4 2 30 30 30 30 0 50 0 0 0 0 0 50 0 50 10 0 0 50 0 20 5 1 10 20 60 0 0 0 0 20 50 30 30 50 20 20 0 10 10 10 20 30 30 0 20 10 20	170 120

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem E
Furniture Factory

There are m workers in Kim's furniture factory. All workers have an equivalent level of skill and productivity. The factory has taken n orders for next month's work. A manager of the factory made a preliminary plan for all orders as follows: For an order i, the earliest start time s_i is the earliest time at which a worker can start working on the order. The working time w_i is the time required to complete the order. The deadline d_i is the time by which the order must be finished. Certainly, $d_i \geq s_i + w_i$.

An order is dealt with by only one worker at any moment. It is possible that a worker stops his current work and works on another order. A suspended work can be resumed by any worker. All workers begin and stop their work at integral times. So, time is numbered 1, 2, 3, … , max$\{d_i\}$, for simplicity.

You, a manager of the factory, are going to make a detailed schedule for next month's production. Given the preliminary plan for the n order requests, write a program to find a schedule such that all orders are finished by their deadline.

Input
Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with a line containing two integers, m and n ($1 \leq m \leq 10$, $1 \leq n \leq 100$), where m is the number of workers and n is the number of orders. In the next n lines of each test case, the i-th line contains three integers s_i, w_i, and d_i (i =1, 2, ... , n and $1 \leq s_i < d_i \leq 500$ and $1 \leq w_i \leq d_i - s_i$), which represent the earliest start time, the working time, and the deadline for an order i, respectively.

Output
Your program is to write to standard output. For each test case, if there is no schedule such that all orders are finished by their deadline, print a single line containing "0". Otherwise, print n lines. The i-th line should contain an odd number of integers "k, a_1, b_1, a_2, b_2, … , a_k, b_k" which represent a schedule for the i-th order, i = 1, 2, … , n, where k means the number of time intervals and (a_j, b_j) means a time interval that a worker will be working on it from a_j to b_j. **Notice that $a_i < b_i < a_{i+1}$.**

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 2 4 2 2 5 1 5 8 4 2 6 3 3 7 1 3 2 4 7 3 1 5 4 2 6	1 2 4 3 1 3 4 5 6 8 1 4 6 2 3 4 5 7 0

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem F
Leet

Leet is an alternative English alphabet used mostly on the Internet. Each character in the English alphabet is replaced by one or more characters which look similar to the original one. For example, the phrase `ACMICPC` can be written as "`4(|V|1(|>(`" in Leet by replacing `A` with `4`, `C` with `(`, `M` with `|V|`, `I` with `1`, and `P` with `|>`. Note that three symbols `|V|` were used to represent a symbol `M` and two symbols `|>` for `P`.

There may be more than one way to represent an English character. For example, `[)`, `|)`, and `|>` may represent `D`. While interesting, Leet can be very confusing and hard to read.

Given one sentence in the Standard English alphabet and another in Leet, write a program to decide whether they are the same or not under the following rules.

a) Each character in the Standard English alphabet is mapped to up to k characters in Leet. The value of k is known in advance.
b) For each character in the Standard English alphabet, there is only one way to represent it in Leet. For example, once `D` is mapped to `[)`, then you cannot use `|>` to represent `D` within the same phrase. But two or more characters of the Standard English alphabet can be mapped to the same sequence of symbol(s) in Leet. For example, both `D` and `P` can be mapped to `|>` simultaneously. An English character and its Leet representation do not need to look similar to each other.

Input
Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case consists of three lines. The first line of each test case contains one integer k ($1 \le k \le 3$) which is the maximum number of characters in Leet for one Standard English alphabet. The second line of each test case contains a phrase in Standard English whose length is between 1 and 15, inclusive. It is written in English lowercase `a-z`. The third line contains a phrase in Leet whose length is equal to or greater than 1. It is written in printable symbols `a-z`, `A-Z`, `0-9`, `@`, `\`, `/`, `-`, `=`, `^`, `|`, `[`, `]`, `(`, `)`, `{`, `}`, <, and >.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain either `0` or `1`. Print `1` if two input strings are the same. Otherwise print `0`.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 3 mississippi nni55i55ippi 2 foobar |=o08ar	1 0

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem G
Laptop

A laptop, also called a notebook, is a personal computer for mobile use. A laptop integrates most of the typical components of a desktop computer, including a display, a keyboard, a CD-ROM drive, etc. Nevertheless, a distinct feature of a laptop is portability, which means that a laptop can be used in many places – not only at home and at the office, but also in coffee shops, in lecture halls and libraries, or at a meeting room, etc. Laptops are typically powered by an internal rechargeable battery that is charged using an external power supply. Typical battery life for standard laptops is two to five hours, but may drop to as little as one hour when doing energy-intensive tasks. A battery's performance gradually decreases with time.

To reduce energy consumption, a laptop has a transition to a standby or hibernate mode if it has been idle for a while. In this mode, no energy is consumed, but a fixed amount of energy, specifically, equal to the energy consumed during a unit time, is required when moving the laptop from the hibernate mode to the active mode. So, minimizing the overall energy consumption is equivalent to finding a schedule to minimize the number of idle periods.

Each task T_i is given with a release time r_i and a deadline d_i, where T_i is executed in a unit time, and it should be scheduled and completed within the time interval $[r_i, d_i]$. Also you can assume the following conditions on the problem instance are satisfied:

(1) r_i and d_i are integers and the tasks T_i must be scheduled at integral times.
(2) $r_i \le r_j$ if and only if $d_i \le d_j$.
(3) There is at least one schedule in which all the tasks can be scheduled and completed within their time intervals.

The goal is to minimize the number of idle periods such that all the tasks are scheduled. Note that the idle periods are assumed to occur after a task is first scheduled.

For example, there are five tasks T_i with release times r_i and deadlines d_i, $i = 1, \ldots, 5$, satisfying that $[r_1, d_1] = [4, 8]$, $[r_2, d_2] = [1, 3]$, $[r_3, d_3] = [8, 10]$, $[r_4, d_4] = [0, 3]$ and $[r_5, d_5] = [6, 8]$. Figure 1 shows a schedule in which the number of idle periods is 3. However, there is a schedule with only one idle period as shown in Figure 2.

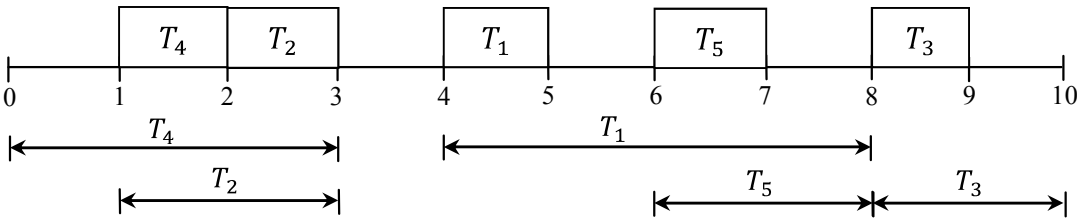


Figure 1. A schedule with 3 idle periods.

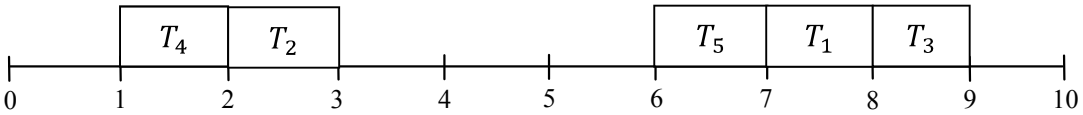


Figure 2. A schedule with the minimum number of idle periods

Input

Your program is to read from standard input. The input consists of *T* test cases. The number of test cases *T* is given in the first line of the input. The first line of each test case contains an integer $n(1 \leq n \leq 100{,}000)$, the number of the given tasks. In the next n lines of each test case, the i-th line contains two integer numbers r_i and d_i, representing the release time and the deadline of the task T_i, respectively, where $0 \leq r_i \leq d_i \leq 1{,}000{,}000$.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line contains the minimum number of idle periods.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 5 4 8 1 3 8 10 0 3 6 8 9 0 4 0 4 3 6 3 6 5 7 6 8 8 11 9 11 10 11	1 0

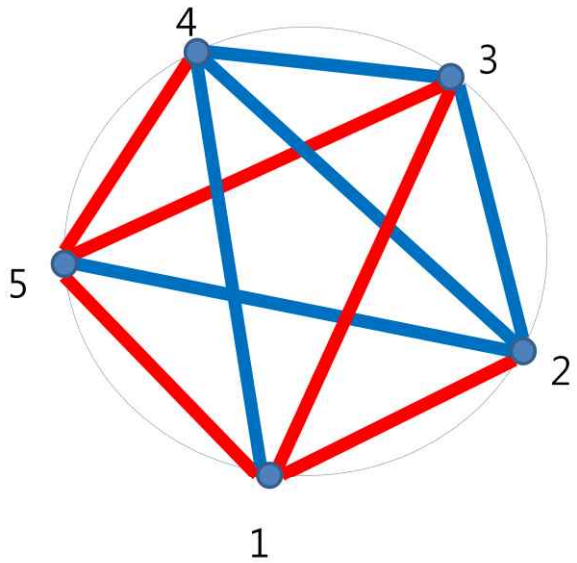
The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem H
Neon Sign

JungHeum recently opened a restaurant called 'Triangle' and has ordered the following neon sign for his restaurant. The neon sign has N corners positioned on the circumference of a circle, and $N * (N-1)/2$ luminous tubes that connect the corners. There are only two colors for luminous tubes, red and blue.

JungHeum wants the sign to show only one shape of a triangle at a time, whose luminous tubes colors are same, continuously. Hence, he wants to know the number of uni-color triangles.

For example, the following neon sign has only two uni-color triangles (1, 3, 5) and (2, 3, 4).



Given the number of corners of the neon sign and the colors of the luminous tubes in the sign, write a program that finds the number of uni-color triangles.

Input
Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case starts with an integer N ($3 \le N \le 1{,}000$), which represents the number of corners of the neon sign. In the following $N-1$ lines, the information about the color of the luminous tubes are given. For the i-th line of these lines, the color information of the luminous tubes that connect corner i to corners from corner $i+1$ to N are given. Note that the color red is represented as 1 and the color blue is represented as 0.

Output
Your program is to write to standard output. Print exactly one line for each test case. The line should contain the number of uni-color triangles.

The following shows sample input and output for two test cases.

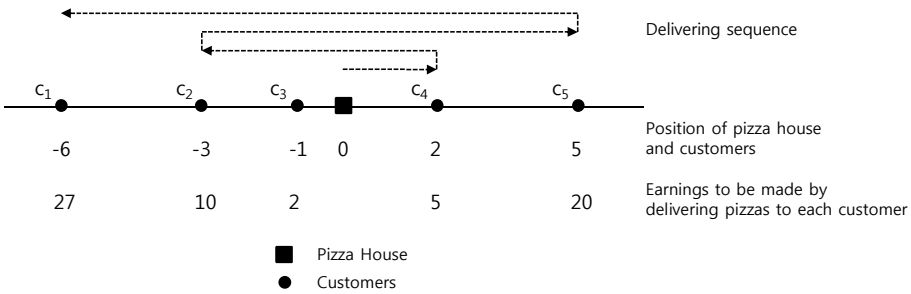
Sample Input	Output for the Sample Input
2 5 1 1 0 1 0 0 0 0 1 1 5 1 1 1 1 0 0 1 0 1 1	2 4

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem I
Pizza Delivery

There is a pizza house located on a straight road, and there are many houses along the road which are customers to the pizza house. To attract more orders from his customers, the owner of the pizza house advertised that he will deduct a penalty for late delivery from the price of the delivered pizzas. The penalty is to be charged when some designated time period has passed after the order has been made, and the amount of penalty to be charged is 1 Korean Won per unit time thereafter. Today all houses on the road ordered pizza at the same time and the delivery of all the ordered pizzas has just started when the late delivery penalty is going to be charged. On a busy day like today, he may not deliver pizza to some customers if the late delivery penalty to be deducted for a customer is more than the earning to be made by selling pizza to the customer. Write a program to help him decide to which customers he has to deliver pizzas and which customers he may skip in order to make the greatest profit. Note that the profit made by delivering a pizza to a customer is the amount of earning from the service deducting the penalty for late delivery. You may assume that his moving velocity is one unit distance per unit time and it takes no time to hand over the pizza to the customer.

For example, the following figure shows the relative positions of five customers $\{c_1, c_2, \ldots, c_5\}$ to the pizza house and the earning to be made by selling pizzas to each customer.



If the delivering sequence of customers is $< c_4, c_3, c_2, c_5, c_1>$, then the amount of penalty for late delivery to each customer is 2 for c_4, 5 for c_3, 7 for c_2, 15 for c_5 and 26 for c_1. In this case the profit from each customer is 3 for c_4, -3 for c_3, 3 for c_2, 5 for c_5 and 1 for c_1. Since the profit from customer c_3 is -3, it is better not to deliver pizza to c_3. Therefore the total profit by delivering pizzas to the customers in this order is 12. The best profit the owner can make, in this case, is 32, where the delivering sequence is $< c_3, c_2, c_1, c_5>$.

Given the relative positions of customers to pizza house, and earnings to be made by delivering pizzas to each customer, write a program to compute the maximum profits by delivering pizzas ordered from the customers.

Input

Your program is to read from standard input. The input consists of T test cases. The number of test cases T is given in the first line of the input. Each test case consists of three lines. The first line of each test case contains an integer, n ($1 \le n \le 100$), which is the number of customers to pizza house. The second line of each test case contains n integers $p_1, p_2, \ldots, p_n$ ($p_1 < p_2 < \cdots < p_n$ and $p_i \neq 0$, $i = 1, \ldots, n$), where p_i is the relative position of

the i-th customer c_i to the pizza house on the straight road. The third line of each test case contains n integers $e_1, e_2, \ldots, e_n$ ($e_i > 0$, $i = 1, \ldots, n$), where e_i is the earning to be made by delivering pizzas to the customer c_i. All integers in the second and third lines are between −100,000 and 100,000 both inclusive.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain the maximum profit by delivering pizzas ordered from the customers.

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 5 -6 -3 -1 2 5 27 10 2 5 20 6 1 2 4 7 11 14 3 6 2 5 18 10 11 -14 -13 -12 -11 -10 1 2 3 4 5 100 200 200 200 200 200 200 200 200 200 200 200	32 13 1937

The 36th Annual

ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem J
Soju

Two rival companies, named IC and PC, share the Soju market in Korea. Soju is well-known Korean distilled liquor, going well with Korean cuisine. Currently they have an agreement on the location of their selling branches; the branches of IC should be located in the west of those of PC, that is, all the IC branches have smaller x-coordinates than any PC branch. At the beginning of every month, they observe any change in the location condition of the agreement. For this, they want to compute the distance of the closest pair of two branches, one from each company. The distance of two points (x_1, y_1) and (x_2, y_2) is defined as $|x_1 - x_2| + |y_1 - y_2|$.

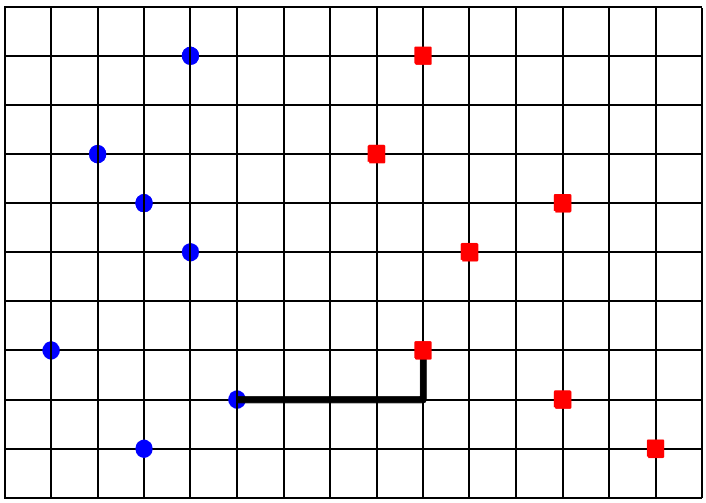


Figure 1. Blue disks are the IC branches and red squares are the PC branches. The distance of a closest pair between them is 5.

You are given two sets of points, I and P, which represent branches of IC and PC companies, respectively. All points in I have smaller x-coordinates than any point in P. You write a program to compute the minimum distance of pairs (i, p) for $i \in I$ and $p \in P$.

Input

Your program is to read from standard input. The input consists of T test cases. T is given in the first line of the input. The first line of each test case contains an integer n, the number of points of I, $1 \le n \le 100{,}000$. Each of the following n lines contains two integers, the x-coordinate and y-coordinate of each point of I. The next line contains an integer m, the number of points of P, $1 \le m \le 100{,}000$. Each of the next m lines contains two integers, the x-coordinate and y-coordinate of each point of P. Coordinates of the points are between -10^6 and $+10^6$ inclusive. Note that all points of I have smaller x-coordinates than any point of P.

Output

Your program is to write to standard output. Print exactly one line for each test case. The line should contain an integer value, the distance of a closest pair of points from I and P.

The following shows sample input and output for two test cases.

Sample Input	**Output for the Sample Input**
2 10 -16 -4 -1 -3 -9 -1 -4 -10 -11 -6 -20 4 -13 6 -3 -10 -19 -1 -12 -4 10 8 2 10 3 10 10 20 -3 20 3 16 2 3 -5 14 -10 8 -2 14 0 10 -3 39 -2 -28 -1 20 -3 11 -3 45 -2 -44 -1 -47 -5 -35 -5 -19 -5 -45 10 27 5 28 0 28 5 21 5 2 3 13 -1 16 -2 20 -2 33 -3 27 1	6 13

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem K
Traveling Spiders

"Traveling Spiders" is a logical puzzle that needs a sequence of systematic choices among many alternatives. The puzzle assumes a geometric object whose surface is partitioned into a set of areas called cells that are usually congruent, or very similar in their shapes and sizes. A spider may move around freely on the surface of the object, but, can move forward from a cell to only one of its neighboring cells in each step. Now, given some pairs of male and female spiders, initially positioned all in different cells, we want to find a set of paths that, respectively, leads each male spider to his partner. The only condition is that every cell of the object must be visited once and only once by a male spider during their traversal.

The "Traveling Spiders Puzzle (TSP for short) on Rubik's Cube" problem considers a Rubik's cube, each of whose six faces is partitioned into $n \times n$ square cells (the above figure exemplifies the case of $n = 4$). Here, two square cells are regarded as a neighbor to each other if and only if they share a side. So, there are four neighbors of a cell, each to east, west, south, and north. Furthermore, we assume that the cube is floating in the air so that all six faces can be traversed by a spider. Note that this puzzle problem is still in its early stage. On the other hand, you are now to write a program to solve a simple single-pair-of-spiders case: given a pair of male and female spiders put in different cells, find a path that passes every cell of the cube exactly once from the male spider to the female spider.

Input

Your program is to read from standard input. The input specifies T test cases. Hence, the value T naturally comes in the first line of the input, followed by a set of three lines, describing each test case. The first line of a test case specifies an even number n, representing the resolution of the cell grid, where we assume that $2 \leq n \leq 50$. The second and third lines indicate the positions of the male and the female spiders, respectively. The locational data of a spider is encoded as coordinates in 3D space under the following rule: a corner or vertex of the cube is regarded as the origin and the directions of the three edges of the cube incident to the origin define those of the positive $x, y,$ and z-axes, forming a right-handed 3D coordinate system. The side of each cell is of length two, implying that the cube exists in the space $[0, 2n] \times [0, 2n] \times [0, 2n]$. Then, the cell that a spider visits is specified as the $x, y,$ and z-coordinates of its center location.

Output

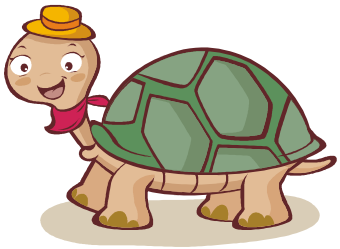
Your program is to write to standard output. For each test case, the first line of the results must contain an integer indicating whether there exists a feasible solution. If yes, the integer must be `1`; otherwise `-1`. If the first line is `1`, then it must be followed by $6n^2$ lines, describing the sequence of cells that the male spider visits on his way to his partner, including the first and last ones. In case multiple solutions are possible, just output any one of them.

The following shows sample input and output for two test cases.

Sample Input	Output for the Sample Input
2 2 1 0 1 3 1 4 2 1 0 1 1 0 3	1 1 0 1 1 0 3 3 0 3 3 0 1 4 1 1 4 1 3 4 3 3 4 3 1 3 3 0 3 1 0 1 1 0 1 3 0 0 3 1 0 1 1 0 1 3 0 3 3 1 4 3 1 4 1 3 4 1 3 4 3 3 3 4 1 3 4 1 1 4 3 1 4 1 1 0 1 3 0 1 4 1 1 4 1 3 4 3 3 4 3 1 3 3 0 3 1 0 1 1 0 1 3 0 0 3 1 0 1 1 0 1 3 0 3 3 1 4 3 1 4 1 3 4 1 3 4 3 3 3 4 1 3 4 1 1 4 3 1 4 3 0 3 1 0 3

The 36th Annual
ACM International Collegiate Programming Contest
Asia Regional - Daejeon

Problem L
Turtle



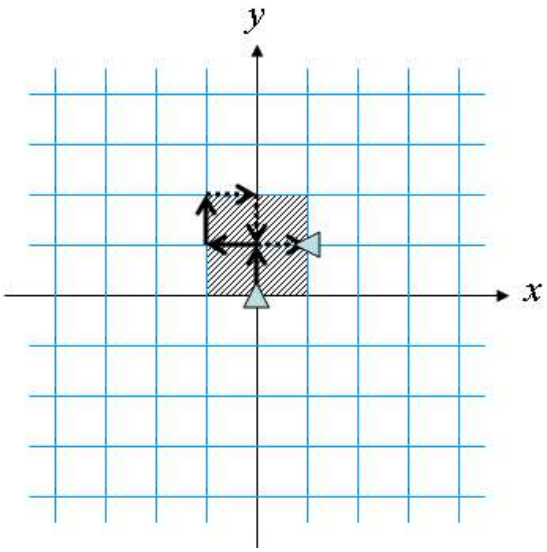
You have a turtle robot moving in the two-dimensional Cartesian plane. The movement of this robot is determined by a sequence of the following commands:

(1) `F`: move forward just one unit
(2) `B`: move backward just one unit
(3) `L`: turn left 90 degree
(4) `R`: turn right 90 degree

The last two commands, `L` and `R`, only change the orientation of the robot without changing its location. We will call the sequence of commands the *control program* of the turtle robot.

You are to determine and estimate a portion of the plane that is large enough for the turtle's movement for a given control program. To make the problem simple, the portion is assumed to be a box whose edges are parallel to either the *x*- or *y*- axis of the Cartesian plane. More specifically, your program should calculate the size of the minimal bounding box which is large enough to encompass the trace of the movements of the turtle for a given control program.

Let's think of an example. You may assume that the turtle is initially at the origin (0, 0) facing to the north, without loss of generality. If the control program is given as `FLFRFLBRBLB`, the turtle will move on the following track of the Cartesian plane:



Note that the dotted portion of the track denotes backward movement. As shown in the picture, the 2×2 gray box is enough to encompass the movement of the turtle. Therefore, your program should write 4, since the width and the height of the bounding box are both 2.

Note that the trace of the turtle may not form a box in some cases. For instance, if the control program is given by `FFLLFF`, the turtle moves only on the *y*-axis. Therefore, the bounding box is not practically a box but a line segment, which implies that the width of the bounding box is zero. You program should output zero in such cases.

Input

Your program is to read the input from standard input. The input consists of *T* test cases. The number of test cases *T* is given in the first line of the input. Each test case consists of a character string where each character denotes the control command. The string is given in a single line and consists only of four commands of the turtle robot. The length of the string is greater than 0 and no more than 500.

Output

Your program is to write to standard output. Print the area of the bounding box as a decimal integer in a single line for each test case.

The following shows sample input and output for three test cases.

Sample Input	Output for the Sample Input
3 FFLF FFRRFF FFFBBBRFFFBBB	2 0 9