ACM-ICPC 2012
Asia Regional Contest
Kharagpur Site

Hosted by
IIT Kharagpur

December 9, 2012

You get:
8 Problems, 25 pages, 300 Minutes

This page intentionally left blank

Rules for ACM-ICPC 2012 Asia Regional Kharagpur Site

1. Solutions to problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judge, and the team is notified of the results. Submitted codes should not contain team or University name and the file name should not have any white space.
2. The public rank list will not be updated in the last one hour. However, notification of accepted/ rejected runs will **NOT** be suspended at the last one hour of the contest time.
3. A contestant may submit a clarification request to judges. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants.
4. Contestants are not to converse with anyone except members of their team and personnel designated by the organizing committee while seated at the team desk. **But they cannot even talk with their team members when they are walking around the contest floor to have food or any other purpose.** Systems support staff may advise contestants on system-related problems such as explaining system error messages.
5. While the contest is scheduled for a particular time length (five hours), the contest director has the authority to alter the length of the contest in the event of unforeseen difficulties. Should the contest duration be altered, every attempt will be made to notify contestants in a timely and uniform manner.
6. **A team may be disqualified by the Contest Director** for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, distracting behavior or communicating with other teams.
7. Team can bring up to 25 pages of printed materials with them. But they are not allowed to bring calculators or any machine-readable devices like mobile phone, CD, DVD, Pen-drive, IPOD, MP3/MP4 players, floppy disks etc. Blank papers will be provided, if needed.
8. With the help of the volunteers and external judges, the contestants can have printouts of their codes for debugging purposes. Passing of printed codes to other teams is strictly prohibited.
9. **The decision of the judges is final.**
10. **Teams should inform the volunteers if they do not get reply from the judges within 10 minutes of submission. Volunteers will inform the External Judges and the external judge will take further action. Teams should also notify the volunteers if they cannot log in into the PC^2 system. This sort of complains will not be entertained after the contest.**
11. Eight problems will be posed. So far as possible, problems will avoid dependence on detailed knowledge of a particular applications area or particular contest language. All problems require you to **read test data from the standard input** as specified in the problem and **write results to the standard output**.

12. You can use any of the standard library functions that your chosen programming language provides. In addition, you can use the math library in C/C++. You cannot use any other library that requires an extra flag to be passed to the compiler command. If you do this, the judges will probably find a code "compilation error" in your program.
13. Your program is not permitted to invoke any external programs. For example, you cannot use in C the system call ("grep xyz abc") to determine whether the file abc contains an occurrence of the string xyz. *Violation of this rule may lead to disqualification from the contest.*
14. **Output must correspond exactly to the provided sample output format, including (mis)spelling and spacing.** Multiple spaces will not be used in any of the judges' output, except where explicitly stated.
15. Your solution to any problem should be submitted for judging using the PC2 software only. Once you have submitted the solution, it will reach the judges. The time it takes for your problem to be judged will depend, among other things, on how busy the judges are. Once your submission has been judged, you will receive a message through PC2 indicating the judgment. The judgment may be "Yes", meaning that your submission was judged to be correct. Otherwise you will get a message indicating the problem with your program. For example, the message may be "Incorrect Output", "Output Format Error", "Compilation Error", "Runtime Error", "Run Time Limit Exceeded" etc.
16. **Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required. The judges will only test whether the input-output behavior of your program is correct or not.**

Problem A: Room Allotment

Input: Standard Input
Output: Standard Output
Problem Code: RoomAllot

Your college is holding the finals of a regional inter-college sports meet. Teams from four regional colleges A, B, C, and D will be arriving in your college soon. It is vacation period and the hostels are empty, so it is decided that they will be put up in your hostel for the duration of the meet. Since the rooms are small, and since the rooms are empty because of the vacation, it is also decided that they will each be allotted a separate room to make their stay comfortable. Each team has K members.

As a member of the organizing team, you have been entrusted with allotting them rooms from a set of 4K side-by-side rooms (numbered 101, 102, 103,…(100 + 4K)) in one long corridor of your hostel. Each room has to be marked as A, B, C, or D, and a team member of college X (X = A, B, C, D) will occupy a room marked X. However, the teams have requested you for some special considerations for their accommodation. Teams A and B have both requested that each member of their team should have another member of their own team in an adjoining room (the room to the left or right). Teams C and D do not trust each other and have both specifically requested that no member of their team should be put up in a room with a member of the other team in an adjoining room.

In how many ways can you mark the rooms?

Input:

First line contains the number of test cases **N**.
Each test case has a single line containing a single integer, the value of **K** ($3 \leq K \leq 100$).

Output:

For each test case, print the case no, followed by a colon, followed by a space, followed by the number of ways the rooms can be marked modulo 1000.

Sample Input	Sample Output
2 4 7	Case 1: 680 Case 2: 520

This page intentionally left blank

Problem B: Travelling Salesmen

Input: Standard Input
Output: Standard Output
Problem Code: Salesmen

Company X sells cosmetic products of different types. In addition to selling their products through stores in larger cities, they also employ a number of salesmen for door-to-door selling in smaller cities. Two salesman are assigned to sell the products in K small cities in a region. Some cities are more lucrative than others in terms of sales potential - each city is associated with an integer profit value estimated from the previous year's sales data. The two salespersons would like to fairly distribute the cities between themselves, so that none of them have to work much more than the other while still having similar profit potential. Hence they would like to distribute the cities such that the number of cities allotted to each do not differ by more than one. In addition, they would like to ensure that the difference between the total profits of the salesmen (sum of the profits of the cities that they visit) is minimized among all such possible distributions. If the profits are not equal, A being senior, gets the cities with higher total profit.

Can you find out the profits that A and B can earn?

Input:

The first line contains the number of test cases **N**.

For each test case, first line of input contains **K**, the number of cities ($0 < K \leq 200$), and the second line contains a sequence of **K** integers with the profit values of each city (profit value of a city is between 1 and 400).

Output:

For each test case, print the case no, followed by a colon, followed a space, followed by the profits of A and B separated by a single space.

Sample Input	Sample Output
2 5 100 210 100 75 340 4 300 250 280 290	Case 1: 415 410 Case 2: 570 550

This page intentionally left blank

Problem C: Loading Machines

Input: Standard Input
Output: Standard Output
Problem Code: MachineLoad

A construction company X has imported a number of large machinery by ship for a project, which have been unloaded and stored in a warehouse in the dock. X employs a fleet of large trailer trucks to move the machinery from the dock to its project site. The dock authorities charge X a storage charge per hour per ton for each machine. Machines are loaded one by one sequentially on the trucks by a single group of laborers, and are moved off to the site as soon as they are loaded. The machines being large, there is a considerable loading time for each machine, and X continues to pay the storage charge for the machine until it is loaded and sent off. The storage charge being significant, X would like to move the machines away to the project site as fast as possible paying the minimum total storage cost to the dock authorities. You can assume that enough trucks are available so that a truck is always available when a machine is sent off and the next one is to be loaded. Can you help company X to decide how to load the trucks and send off the machines?

Input:

The first line contains the number of test cases **N**.

For each test case, first line contains two integers, the number of machines **K** ($0 < K \leq 200$) and the storage charge **C** per hour per ton. Second line contains a sequence of **K** real numbers (upto 2 decimal place maximum) indicating the weight in ton of each machine. Third line contains a sequence of **K** real numbers (up to 2 decimal place maximum) indicating the loading time in hour for each machine.

Output:

For each test case, print the case number, followed by a colon, followed by a space, followed by the total storage cost paid by X rounded off to the nearest integer.

Sample Input	Sample Output
2 5 3 7.5 6.5 5 4.25 3.5 1.5 1.45 1.7 1.6 2 4 7 20.12 15.75 12 8.4 5.4 5 4 3.5	Case 1: 327 Case 2: 4169

This page intentionally left blank

Problem D: Antakshari

Input: Standard Input
Output: Standard Output
Problem Code: Antakshari

Antakshari is a game that can be played by two or more teams, in which the teams take turns to sing songs. The first team starts the game by singing a song. The next team at each step has to sing a song that starts with the last letter of the song sung by the previous team, which has not been sung before by any team so far. A team that cannot find a matching song at any step when its turn comes up is eliminated. The game continues until a single team is left.

Two friends A and B would like to play Antakshari. However, since they have played Antakshari several times before, they decide to try out a variation just for fun. The songs are to be chosen from a common pool of songs that is known to both of them. Two songs are of the same type if they have both the same first letter and the same last letter. For the first move, Player A chooses any song from the pool. The players alternate their moves. During its turn, (other than for the first move) a player must choose a song from the pool which has not been chosen yet, and whose first letter matches the last letter of the previous song. The game ends when there is no song in the pool which begins with the last letter of the previous song, and which has not been chosen yet. The player who was the last to choose successfully wins in this case.

B graciously asks A to start the game. A can start with any song, but would like to start with a song that increases his chance of winning depending on the choices that B makes in the game. But that seems difficult, given that A does not know what choices B is going to make as the game has not even started.

A winning strategy for A is a choice of the first song which will guarantee that A will win irrespective of the choices that B makes subsequently at any step of the game. Can you find the number of songs in the pool for which A has a winning strategy?

Input:

The first line contains the number of test cases **N**.

For each test case, the first line contains two integers, number of songs **K** ($0 < K \leq 150$) and number of types **T** ($0 < T \leq 6$), $T < K$. This is followed by **K** lines, each line containing one song (a string of characters of maximum length 50)

Output:

For each test case, print the case no., followed by a colon, followed by a space, followed by the number of songs for which A has a winning strategy (if there is no such song, print 0).

Sample Input	Sample Output
2 8 4 a night in the woods and There Were Shepherds an Old Fashioned Christmas all you need is elves silver anklets stories of ghosts step into durga puja a walk in the mountain 7 3 a night in the woods and There Were Shepherds an Old Fashioned Christmas all you need is elves silver anklets stories of ghosts step into durga puja	Case 1: 5 Case 2: 4

Problem E: Choosing Volunteers

Input: Standard Input
Output: Standard Output
Problem Code: Volunteer

An NGO would like to man a community helpline 365 days a year (yes, they get the last day off on leap years!). Not having too much money and resources, they advertise for volunteers who will be willing to offer their services for some nominal fee for the next year. They were surprised to get a large number of offers in response to the advertisement. In fact, someone or other has volunteered to work for each day of the year! Each volunteer has offered to work for some number of consecutive days (fixed dates, like 3rd to 5th April, 12th to 21st June, etc.) for a nominal amount of money to cover their basic expenses like travel, food etc. Different volunteers may have to travel from different places and may have offered their services for different number of days, and hence their travelling expenses etc. may be different. Hence the money that each volunteer asks for may vary.

The NGO is thrilled with so many offers, but now has to choose some of the volunteers. Having a choice, and with little money at its disposal, it would like to choose a set of volunteers such that on each of the 365 days, there is at least one volunteer present, while minimizing the total amount of money it has to spend. For each offer, it will have to accept the offer fully (paying the full amount) or reject it fully (paying nothing), it cannot accept an offer for only some of the days in the offer. You have to help the NGO to find a set of volunteers to accept.

Input:

First line contains the number of test cases **N**.

For each test case, first line contains the number of offers **K** ($0 < K \leq 500$). The next **K** lines each contains 3 integers, the starting day of the offer (an integer from 1 to 365), the ending day of the offer (an integer from 1 to 365), and the amount of money asked (an integer). The starting and ending day are specified assuming that the days in a year are numbered from 1 to 365. For all offers, starting day ≤ ending day.

Output:

For each test case, print the case no., followed by a colon, followed by a space, followed by the total amount of money that the NGO has to spend.

Sample Input	Sample Output
2 5 1 100 200 50 180 100 1 200 500 150 365 200 90 365 1000 6 1 100 70 101 200 150 201 300 160 301 365 80 75 150 50 151 300 100	Case 1: 500 Case 2: 300

Problem F: Social Game

Input: Standard Input
Output: Standard Output
Problem Code: GroupFriends

In order to foster interaction among the children, a school decided to divide their students into two groups and take them on a day-long excursion on two different days with different group activities planned. However, to promote interactions between diverse groups, it decided to form the two groups such that for any student X in one group, at least half of his/her friends are in the other group. Thus, if X has P friends, at least $\lceil P/2 \rceil$ of his friends are in the other group. The school knows the friends of all students (assume that X is a friend of Y means Y is also a friend of X) and will have to form the groups. However, given the large number of students and their friend lists, the school is finding it very difficult to do this, and would like you to help. The group sizes may be skewed and each group may have any number of students, that is not a problem for the school, the focus is to ensure that the students interact with other students who are not their friends.

Can you tell the school how to divide the students into the two groups?

Input:

The first line contains the number of test cases, **N**.

For each test case, the first line has the number of students **K** ($0 < K \leq 1000$), followed by **K** lines, one for each of the students. Each of these lines contains an integer student id (the students are numbered from 1 to **K**), followed by the number of friends **P** the student has, followed by **P** integers for the student ids of the **P** friends. Note that some student may have no friends, in which case the line will contain 0 for **P** with no following integer (for example. just "23 0" if student with id 23 has no friends).

Output:

For each test case, print the case number, followed by a colon in the first line. The second line prints the number of students **M** in the first group, followed by a space, followed by the student ids of the **M** students in the group, sorted by increasing student id with one space between successive ids. The third line prints the same for the second group. You can print the groups in any order.

Sample Input	Sample Output
2 5 1 3 2 3 4 2 4 1 3 4 5 3 2 1 2 4 3 1 2 5 5 2 2 4 5 1 3 2 3 5 2 3 1 4 5 3 3 1 4 5 4 3 2 3 5 5 4 1 2 3 4	Case 1: 2 1 5 3 2 3 4 Case 2: 3 2 3 5 2 1 4

Problem G: Farmhouse Fencing

Input: Standard Input
Output: Standard Output
Problem Code: FarmFencing

Mr Firm has four strange farmhouses whose walls are all vertical, rectangular, and 2 meters high. But the strange part is that each farmhouse has a very large number of walls, and the walls have arbitrary orientations. As a result, the ratio of wall-area to floor-area of each farmhouse is arbitrarily large. The four farmhouses are located in four adjacent equal-area square *farmlands* so that each farmland contains one farmhouse only, and the union of the four square farmlands is a square which is 4 times as large as each square farmland. It is assumed that sides of the square farmlands are parallel to *x*- and *y*-axes.

Mr Firm wants to fence his four farmhouses by a *single fencing* of height 2 meters so that the four farmhouses lie within the common fenced area. Note that the new fencing may include some wall of the old fencing and some corner of a farmhouse may touch the fence. By definition, each *corner* is defined by the intersection of two consecutive walls.

The cost of fencing is Rs. 50 per square meter. Mr. Firm would like to find the least cost (rounded to nearest integer) that he has to spend for the fencing. Can you help Mr. Firm to find the money he will have to spend for fencing his farmlands?

Input:

The first line contains the number of test cases, **N**.

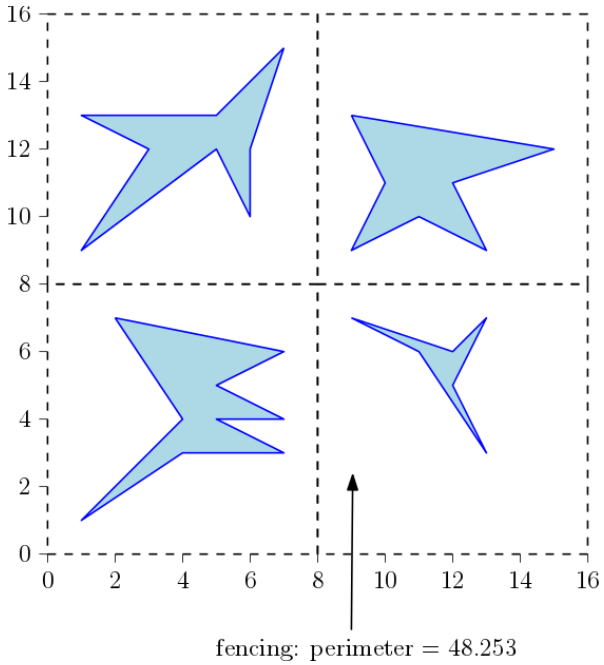
For each test case, there are four lines. The 1^{st} line contains the number of corners **C** ($3 < C \leq 20000$) of the top-left farmhouse, followed by (*x, y*) coordinates of its corners, starting from the leftmost corner (i.e., minimum *x*-coordinate). If two (or more) corners have the same minimum *x*-coordinate, then the one with minimum *y*-coordinate among them appears first. The corners are specified either in clockwise or in anticlockwise order. The 2^{nd} line is for similar information corresponding to the top-right farmhouse, and the 3^{rd} and the 4^{th} lines correspond to the bottom-left and the bottom-right farmhouses respectively.

Output:

For each test case, print the case no., followed by a colon, followed by a space, followed by the cost of fencing rounded to the nearest integer.

Sample Input	Sample Output
1 8 1 9 3 12 1 13 5 13 7 15 6 12 6 10 5 12 7 9 9 10 11 9 13 15 12 12 11 13 9 11 10 9 1 1 4 4 2 7 7 6 5 5 7 4 5 4 7 3 4 3 6 9 7 12 6 13 7 12 5 13 3 11 6	Case 1: 4825

The sample input above is for the farmhouses (shaded regions) shown below. The minimum total length of the single fencing will be 48.253m as indicated.



Problem H: Pattern Fracturing

Input: Standard Input
Output: Standard Output
Problem Code: PatternFracture

You are given a simple (non-self-intersecting) polygon with integer coordinates. The sides of the polygon are aligned with 0°, 45°, 90°, or 135°. That is, the sides are horizontal, vertical, or diagonal. A sample polygon is shown in Figure 1.

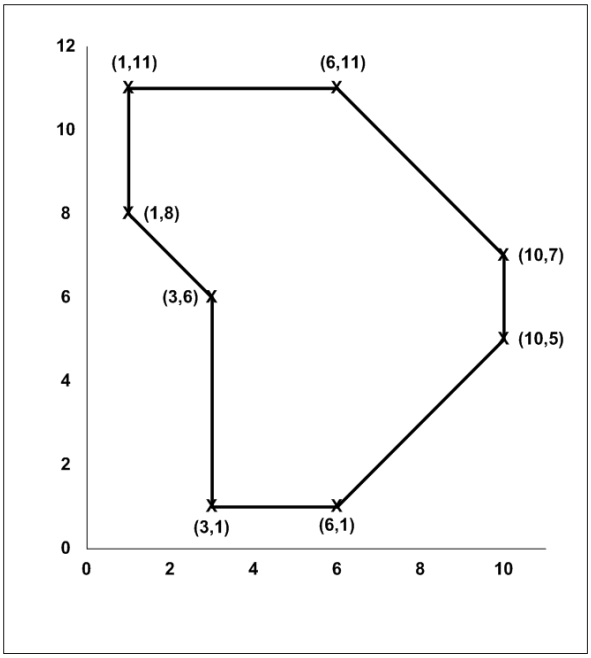


Figure 1: Polygon to Fracture

You are also given a set of 27 primitive patterns as shown in Figure 2. The properties of the patterns are listed in Table 1. Every pattern is specified by its width (**w**) and height (**h**). Rest of the dimensions follows from the geometry (the fact that sides must be horizontal, vertical or diagonal). Every pattern is assumed to be contained in a minimal Isothetic (axis aligned) rectangle. The left-bottom corner of this rectangle is taken to be the center of the pattern. For tiling (to be discussed below), use the center to uniquely specify the position of a pattern. Any pattern placed anywhere on the integer plane can be specified by a triplet: **<(x, y), c>** where (x, y) is the center and c is the code of the pattern (Figure 2, Table 1).

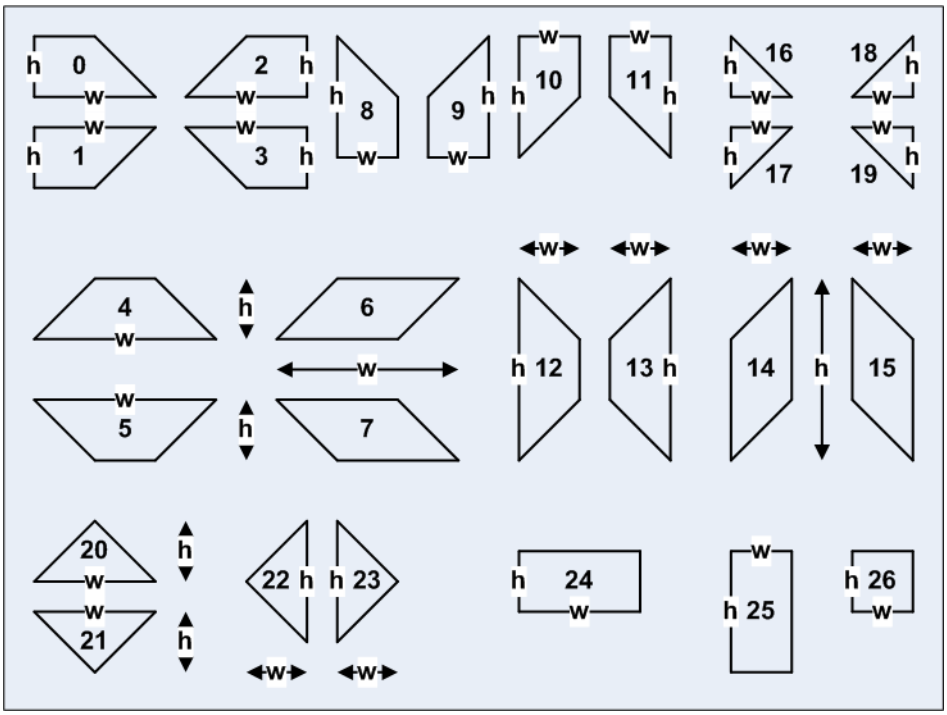


Figure 2: Tiling Patterns

Table 1: Properties of Tiling Patterns

Code	Width	Height	Code	Width	Height	Code	Width	Height
0	2	1	9	1	2	18	1	1
1	2	1	10	1	2	19	1	1
2	2	1	11	1	2	20	2	1
3	2	1	12	1	3	21	2	1
4	3	1	13	1	3	22	1	2
5	3	1	14	1	3	23	1	2
6	3	1	15	1	3	24	2	1
7	3	1	16	1	1	25	1	2
8	1	2	17	1	1	26	1	1

You need to tile the polygon using the patterns. Every tiling should exactly cover the polygon. That is, there should be no area not covered by a pattern and no pattern or part of it should go beyond the boundary of the polygon. Patterns cannot overlap – they touch each other along their respective borders. Whenever two patterns touch along their borders, they are assumed to have been stitched along it.

You may use any number of patterns of any type (Figure 2) in tiling the polygon. A sample tiling is shown in Figure 3.

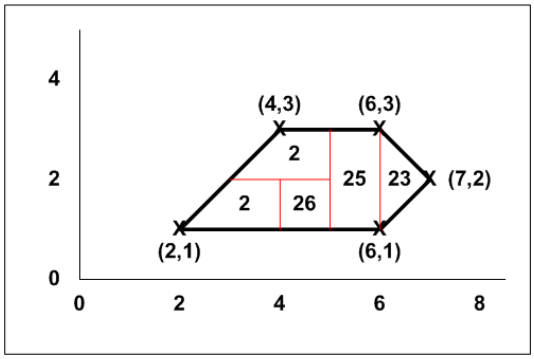


Figure 3: Sample tiling of a polygon

Every tiling has a cost computed as follows:

Cost C(P) of tiling a Polygon P = Total length of horizontal stitches
+ Total length of vertical stitches
+ Total number of diagonal sides in the tiling patterns used

The stitches are computed between pairs of patterns only. Hence no stitch is counted where the border of a pattern touches the border of the polygon. In Figure 3, C(P) = 7. This is computed by counting the red links of unit length.

Note that the total number of diagonal sides in the tiling patterns used is **not the same** as the number of diagonal sides in the stitch, as a single long diagonal side in the stitch may have diagonal sides from more than one tiling pattern.

A polygon can have tilings with different costs. For example for the polygon P (Figure 1), Figure 4 and Figure 5 present two alternate tilings with costs C(P) = 115 and C(P) = 86.

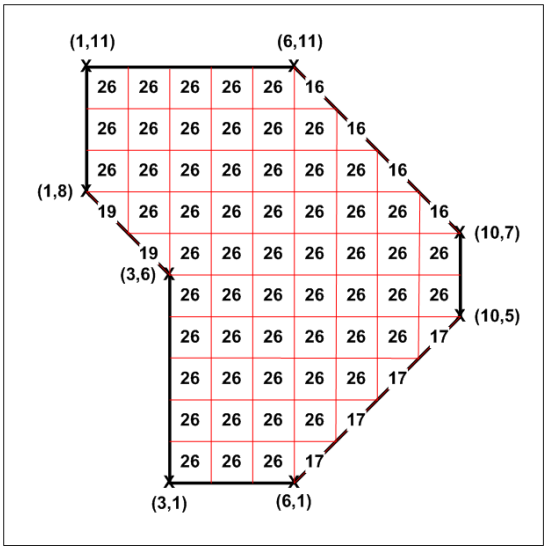


Figure 4: Tiling of polygon in Figure 1 with Cost = 115

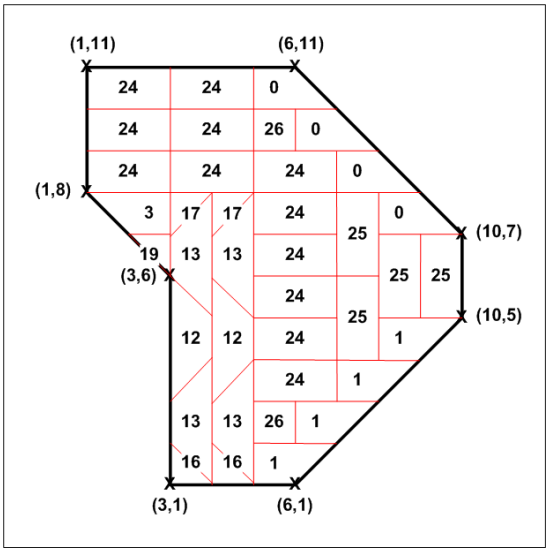


Figure 5: Tiling of polygon in Figure 1 with Cost = 86

Given a polygon P, you want to find a tiling that minimizes C(P). However, finding the exact minimum is hard. So you decide to do as well as you can and find a tiling that reduces the cost.

Input:

The first line contains the number of tests cases **N**.

For each test case, the first line contains the number of vertices **M** ($0 < M \leq 10{,}000$) of the polygon to be tiled. Each of the next **M** lines contains the x and y coordinates of a vertex of the polygon. The vertices are listed in the counterclockwise order starting from the bottom-most left-most vertex of the polygon.

Output:

For each test case, print the case number, followed by a colon, followed by a space, followed by the cost of the solution in the first line. The second line will contain the number of patterns **P** used in the tiling. Each of the following **P** lines lists the patterns (specified as triplets) in the tiling in the bottom-to-top, left-to-right order.

For the exact format, see the sample input and output given next. In the sample input, the first test case is the polygon in Figure 3. The sample output for Case 1 shows the tiling shown in Figure 3. The second test case is for the polygon shown in Figure 6.

Sample Input	Sample Output
2 5 2 1 6 1 7 2 6 3 4 3 13 1 0 2 0 2 3 3 3 2 4 3 5 2 5 2 6 1 5 0 5 1 4 0 3 1 2	Case 1: 7 5 2 1 2 4 1 26 5 1 25 6 1 23 3 2 2 Case 2: 5 5 1 0 25 0 2 3 0 3 4 0 4 5 1 5 18

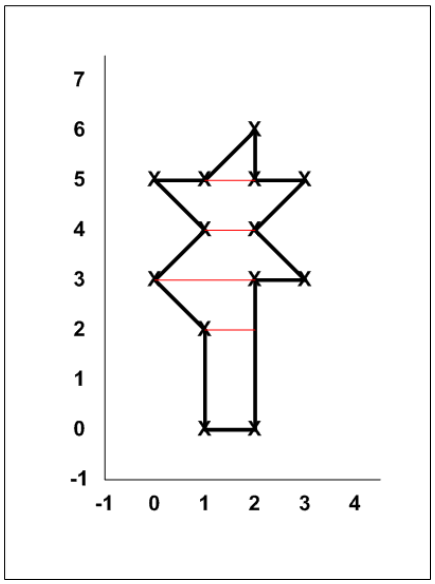


Figure 6: Polygon for 2nd test case in Sample Input

To help you in estimating the quality of solutions you get, we have given you a sample input file ***SampleInput.txt*** with three test cases in the above format. The cost of a "good" tiling for the three cases is shown in the table below. You solution should aim to achieve a tiling with cost equal to or lower than that. Note that this does **not** imply that your code, when submitted, will pass the actual test cases.

Test case no. in *SampleInput.txt*	**Cost of a "good" tiling**
1	≤ 100
2	≤ 3400
3	≤ 7500

END OF PROBLEMS