Benelux Algorithm Programming Contest
Problemset

October 25, 2008

A Song contest

Every year, the continent of Cacophonea organises the Cacophonean Song Contest, for which each of its nations presents an act by a national singer or group. Each Cacophonean inhabitant may televote for any act which is not from his nation, so a nation can never vote for its own act. In the end, each of the s Cacophonean nations will award points to r acts. The act which received most votes from a certain nation will get r points from this nation, the act with the second largest amount of votes will get $r-1$ points, etc., so the act with the rth largest amount of votes gets 1 point and less popular acts get no points from the voting nation. The final ranking of the song contest will then be decided by the total amount of points each nation received.

Music producer Dustin has been eagerly following the Cacophonean Song Contest for many years. Lately, he has observed that in certain nations, televotes are politically rather than artistically motivated:

- Politically voting nations prefer acts from neighbouring nations. More specifically, the Euclidean distance between their capital and another nation's capital is their popularity measure, irregardless of the artistic quality of the corresponding act. Hence, the nation with the closest capital will get most votes and the nation with the farthest capital will receive the least votes, which could yield no points at all if $r < s-1$. It will never occur that two or more capitals will have the same distance to a certain capital.

- Nations that vote quality-motivated will, as the name suggests, award points to nations according to an indisputable act ranking based on each act's artistic quality. In this ranking, there will be no ties so each nation has its own unique rank.

However, Dustin has heard he can influence the voting behaviour of other nations: an artist can find favour of politically voting nations by giving them special attention during his act (e.g. by singing parts in their local dialects, waving their flags, etcetera). The more a politically voting nation will receive such attention in an act, the higher it will rank it. Of course this will be at the cost of the original act and might result in terribly campy results. Therefore, nations that vote according to artistic quality will punish such behaviour.

More specifically, Dustin can divide an act into exactly $s-1$ parts. Originally, all parts are dedicated to the nation of the performer (i.e. they reflect his original artistic ideas), but this can be changed:

- For each part in the act that will be dedicated to a certain politically voting nation, that nation will rank the performer's nation one place higher (unless the performer's nation is already ranked first). As each nation has a unique ranking position, the nation that originally was at that higher rank will then be ranked one place lower.

- Quality-motivated voting nations do not like these soft-soaping attempts at all. Therefore, for each part in the act that will be dedicated to a nation which is not the original performer's nation, quality-motivated nations will rank the original performer's nation one place lower (unless the performer's nation is already ranked lowest).

Only the fact that a certain amount of parts is dedicated to a certain nation will influence voting behaviour: the exact part dedication sequence in the total act is not of interest here.

Dustin wants to use this knowledge (which no other Cacophonean producer has) to produce a smashing act, yielding as much points in the overall results as possible. You are asked to determine what the largest possible overall point score is he can obtain for an act, when he optimally exploits the described act-changing practices.

A.1 Input

- The first line of input consists of the integer number n, the number of test cases;

- Then, for each test case:

- A line with an integer number s ($1 < s \leq 100$), indicating the number of participating nations in the song contest;
- Then, for each nation:
  - A line containing:
    - A string c (not containing any spaces) with the nation's name, followed by a space. Within a test case, there will not be multiple nations sharing the same name;
    - A character indicating the nation's voting behaviour: 'q' if the voting behaviour is quality-motivated and 'p' if the behaviour is politically motivated.
  - A line containing:
    - The location of the nation's capital, expressed in an (x, y) integer coordinate ($-10000 \leq x \leq 10000$, $-10000 \leq y \leq 10000$). x and y are separated by a space. Furthermore, y is followed by a space;
    - The actual artistic quality rank q of the nation's act. This is a unique number in the range $1 \ldots s$.
- A line with an integer number r ($0 < r \leq s - 1$), indicating to how many nations each nation will attribute points;
- A line with the name of the nation for which Dustin should produce a song, achieving as much points as possible.

A.2 Output

For each test case, the output contains a single line with a single integer number: the maximal amount of points an act can obtain in the overall final score, if act-changing practices were performed in an optimal way.

A.3 Example

Input	Output
2 3 Aulatrias q 0 0 1 Binen q 5 0 2 Cahin q 0 -4 3 2 Cahin 3 Aulatrias p 0 0 1 Binen p 5 0 2 Cahin p 0 -4 3 2 Binen	2 4

Table 1: Example input and output

B Message

When little Sascha grew up, she lost her bad habit of pronouncing words in ways that were most easy for her, but did not always match the correct pronunciation. However, she never lost the linguistic creativity she used to show as a little girl. When the Earth Allied Forces (EAF) discovered she also had brilliant mathematical insights and a knack for puzzles and secret messages, she was immediately offered the position of Head of the EAF Intelligence Department.

Sascha's current task is interpreting intercepted internal messages of the hostile Mars Federation. Although Martian messages always consist of just one word, her task turns out not to be easy, as two factors influence the content of the intercepted message:

- Extraterrestrial environment conditions are so bad that errors can occur in intercepted messages, causing them to be quite obfuscated compared to the originally sent text. If such errors occur, the erroneous characters will be characters from the Martian alphabet, just as the original characters.
- Furthermore, linguistic characteristics play an important role. In Martian, there are relations between two subsequent characters: for a given character, some characters are more likely predecessors than others (note that something similar occurs in English: for example, a 'h' in a word will more likely have been preceded by a 't' than by a 'q').

Fortunately, probabilities that a received character y actually was sent as an original Martian character x is known for all alphabet characters, as well as the probabilities that a certain character x_i occurs in a clean Martian word if it was preceded by a Martian character x_{i-1}.

Given all these probabilities, Sascha wants to find the so-called *maximum likelihood* text for a received message, which is the most likely message the Martians originally sent. As senior programmer in the EAF Intelligence Department, you must write a program for her, achieving this goal for several intercepted messages in several local Martian dialects.

To give a simple example, consider a local Martian alphabet only consisting of the characters 'a' and 'b' and let the receiving error probabilities and character succession probabilities be as shown in Table 2. If the intercepted message just consists of an 'a', this can either originally have been an 'a' or a 'b'. With no previous characters available, only the error probabilities are considered: it then turns out that the maximum likelihood message is an 'a' as well, with probability 0.9.

	Received Character o_i	
True Character m_i	a	b
a	0.9	0.1
b	0.1	0.9

	Current Character m_i	
Previous Character m_{i-1}	a	b
a	0.8	0.05
b	0.2	0.95

Table 2: Example Receiving Error Probabilities (left) and Character Succession Probabilities (right).

To extend the example, if the intercepted message was 'ab', we also need the character succession probabilities. The probability that the original message was 'aa' is

$$\begin{gathered} p(\text{received 'a' indeed was originally sent as 'a'}) \\ \times p(\text{received 'b' was originally sent as 'a'}) \times p(\text{character 'a' succeeds previous 'a'}) \\ = 0.9 \times 0.1 \times 0.8. \end{gathered}$$

Similarly, the probability that the original message was 'bb', 'ab' or 'ba' are $0.1 \times 0.9 \times 0.95$, $0.9 \times 0.9 \times 0.05$ and $0.1 \times 0.1 \times 0.2$, respectively. Hence, the maximum likelihood message now is 'bb'. In all cases asked for, there will always be a unique maximum likelihood message.

B.1 Input

- The first line of input consists of the integer number n, the number of test cases;
- Then, for each test case:
  - An integer number a ($0 < a < 30$), which is the number of characters in the local Martian alphabet;
  - A line containing the a unique characters $c_1, c_2, \ldots, c_a$ of the local Martian alphabet, separated by single spaces. Whitespace characters will not occur in the alphabet;
  - a lines, specifying receiving error probabilities for the a alphabet characters in the order in which they were presented. The ith line corresponds to the error probabilities for the original alphabet character c_i and contains:
    - a floating point numbers $e_{i1}, e_{i2}, \ldots, e_{ia}$, separated by single spaces. Number e_{ij} ($0 \leq e_{ij} \leq 1$) indicates the probability that an observed character c_j originally was sent as the character c_i (hence $\Sigma_{i=1}^{a} e_{ij} = 1.00000000$);
  - a lines, specifying character succession probabilities for the a alphabet characters in the order in which they were presented. The ith line corresponds to the character succession probabilities for cases where the original alphabet character c_i was the immediate predecessor and contains:
    - a floating point numbers $s_{i1}, s_{i2}, \ldots, s_{ia}$, separated by single spaces. Number s_{ij} indicates the probability that a certain character c_j has character c_i as immediate predecessor (hence $\Sigma_{i=1}^{a} s_{ij} = 1.00000000$);
  - An integer number w ($0 < w < 50$), indicating the number of intercepted messages in the local Martian alphabet specified;
  - Then, for each intercepted message:
  - A line containing the intercepted message. These messages are non-empty, case-sensitive and will not exceed 300 characters in length.

Given floating point numbers never have more than 10 decimal numbers following the separator '.'.

B.2 Output

For each message in each test case, the output will consist of a single line with a single string: the maximum likelihood original Martian message given the intercepted message.

B.3 Example

Input	Output
1 2 a b 0.9 0.1 0.1 0.9 0.8 0.05 0.2 0.95 2 a ab	a bb

Table 3: Example input and output

C The skatepark's new ramps

The local skating park has been given a financial incentive by the city to make the park interesting for skaters of all levels. The park wants to use the incentive to build a series of ramps, somewhat resembling a mountain range. When talking to some of the volunteers in the committee responsible for the project, you find out they're having difficulties deciding about the best configuration of the ramps. They know the number of ramps to be built, and for each ramp they agree on the range of the height for that ramp. They are still discussing exactly how high each ramp should be, since they can't afford to have them all at their highest, but they do want to spend all of the budget. This is the most important issue in the debate: they can't agree whether they want the differences between the ramps to be small, to give the full ride a more consistent feeling, or as big as possible, to create a more diverse set of challenges.

You also notice they don't really have a good idea what the possibilities are, leaving them stranded in 'what-if' discussions. You decide to help them out by showing them the options they have, both the ones where the difference between the highest and lowest ramp is kept as small as possible, as well as the one where that difference is as much as possible. Since the committee is mainly bickering over the allowable differences, you decide to start out by just presenting them the minimum and maximum difference between the highest and lowest ramp. Luckily, the park has a lot of space, so you won't need to take the placement of the ramps into account. All ramps have the same inclination, which is such that a ramp of height h will have a length $4h$ (measured flat, not over the ramp).

C.1 Input

- The first line of input consists of the integer number n, the number of test cases;
- Then, for each test case:
  - A line with the integer number r ($2 \leq r \leq 10000$), the number of ramps the park will place;
  - A line with the integer number m ($0 \leq m \leq 200000000 = 2 \times 10^8$), the number of cubic meters of concrete the park has money for;
  - r lines with two numbers, l and t ($0.00 \leq l \leq t \leq 100.00$), separated by one space, the minimum and maximum height in meters of the r-th ramp.

You may assume all ramps are made entirely of concrete, and shaped as 1 meter wide prisms, with a triangle with two equal sides as base. A series of ramps within the given constraints and using all concrete is guaranteed to exist.

C.2 Output

For each test case, the output contains one line with two numbers, separated by one space: the minimum difference between the highest and lowest ramp and the maximum difference between the highest and the lowest ramp. These numbers are rounded to two decimals.

C.3 Example

A configuration of the ramps for the example with a maximum difference between the highest and lowest ramp is shown in figure 1.

Input	Output
1 3 36 1.00 4.00 1.00 4.00 1.00 4.00	0.00 3.00

Table 4: Example input and output

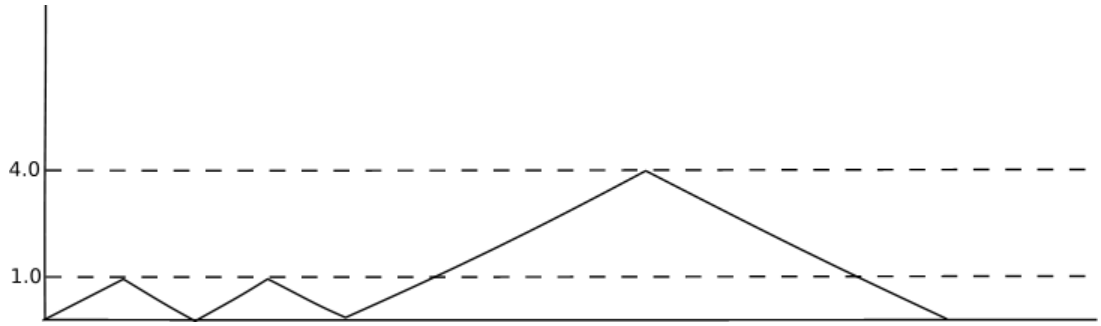


Figure 1: The example ramps with maximum difference.

D Road

In the Green Heart of Holland the villages are small and the roads narrow. Some roads are only one car wide, so if two cars travelling in the opposite direction meet on such a road, they have a problem. In the Green Heart of Holland there usually is a canal at each side of the road, which makes it is impossible for a car to leave the road to let another car pass. To solve this problem, the roads here and there are made a bit wider, so one car can get aside, and one or more cars from the other side may pass. This works fine, as long as the traffic density is low. If, within a short period of time, a lot of cars from both sides try to pass the road, the traffic will get stuck. Given the number of cars entering at both sides (and the location of the passing places) there must be an optimal schedule, to have them all pass the road as soon as possible. Finding such a schedule may be quite hard, so we will not ask you to do so. We ask you to solve a more simple problem.

Given is a road (running East-West), with its passing points, a number of eastbound cars and a number of westbound cars. (An eastbound car enters the road at the West side). Furthermore a schedule is given, that is, for every pair (e,w), where e is an eastbound car and w is a westbound car, the place where they pass each other is given. If the cars enter the road with long delays, such a schedule may take an arbitrary long time. We assume, however, that all cars are ready to enter the road, and eager to leave the road as soon as possible. We measure the time between the moment the first car enters the road, and the moment the last car leaves the road, and we want that interval to be as short as possible. All cars either stand still, or are driving with a constant speed of 45 km/h. Starting and stopping does not take any time. Two cars driving in the same direction always keep a distance of at least 25 meter. The distance between two different passing places is at least 30 meter. The length of the cars may be ignored.

D.1 Input

- The first line of input consists of the integer number n, the number of test cases;
- Then, for each test case:
  - A line containing the length l ($0 < l \leq 30000$) of the road in meters, and the number p ($0 < p$) of passing places;
  - A line containing p positive integers, the distance in meters from each passing place to the west end of the road. These numbers are sorted in increasing order;
  - A line containing two positive integers e and w, ($0 < e, w \leq 1000$) the number of eastbound and westbound cars respectively;
  - e lines, containing w numbers each, indicating the passing points of each pair of cars. The number z on position x ($1 \leq x \leq w$) of line y ($1 \leq y \leq e$) indicates that the eastbound car y passes the westbound car x at passing point z ($0 \leq z \leq p + 1$). Westbound car x enters the road before westbound car $x + 1$, and eastbound car y enters the road before eastbound car $y + 1$. The value $z = 0$ indicates crossing at the west end, so car y enters the road after x has left the road. The value $z = p + 1$, in the same way, indicates crossing at the east side.

  All numbers on a line are each separated by one or more spaces.

D.2 Output

For each test case, the output contains one line with one number: the time (in seconds, rounded to the nearest integer) it takes to have all cars pass the road.

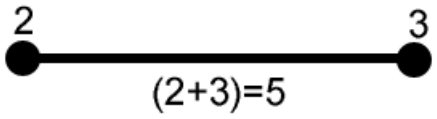
D.3 Example

Input	Output
2 150 1 50 1 1 1 100 1 30 3 2 2 2 1 2 0 2	16 48

Table 5: Example input and output

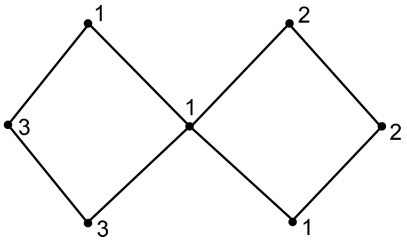
E Warfare

In the year 2240 a great war is taking place between the Earth Allied Forces (EAF) and the Mars Federation (MF). Until recently, none of the two major factions could gain the upper hand in the war. Because of a recent financial crisis, both factions' resources are thinning out, and it appears that the MF is using this to their advantage to claim more territories over the EAF. In response to this, the EAF have decided to carry out its greatest operation since the start of the war: it will launch a simultaneous attack against all the MF bases that are scattered over the planet Mars. The EAF's forces mostly consist of mechs, which are huge bipedal, limbed vehicles with flying capabilities.
A typical MF base is built up as follows: The buildings that make up the base are positioned over one or multiple territories. Each of these territories is protected against outside attacks by impenetrable energy fields that are generated by shield towers. These shield towers are positioned around the territories they're supposed to protect.
Each shield tower is connected to at least one other tower via channels that are constructed above the ground. When these connected towers form a cycle, they generate an energy field. However, if a channel in a cycle is destroyed so that the cycle is broken, the energy field will disappear.
The MF knows that if all energy fields disappear, the base will be easily overrun by the EAF forces. Therefore, the two towers that are connected by a channel protect the channel against armed forces. Each tower has defenses that can take down a given number of enemy mechs. Each channel can handle an attack by a certain number of enemy mechs before it collapses. This number is given by the amount of mechs that both towers, that the channel connects, can take down. Two towers cannot be connected by more than one channel.

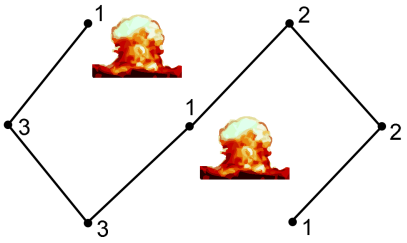


(a) Two towers that are connected by a channel. The vertices represent the towers while the line is the channel that connects the towers. The amount of mechs needed to destroy the channel is the combined amount of mechs that the connected towers can take down.

However, attacking a channel on one side of the tower does not diminish the amount of mechs that it can take down on the other side of the tower.
Because the operation is a surprise attack, all attacks on the channels must happen simultaneously: all channels are taken down at the same moment.



(b) MF base with multiple energy fields. The vertices represent the towers while the lines are the channels that connect the towers. The numbers indicate how many mechs a tower can take down.



(c) In this case, the destruction of two channels will make all energy fields disappear. Four mechs will be lost in the battle.

All energy fields must be disabled in order to destroy a MF base. Tearing down all channels would make this happen, but would also require a lot of mechs to be sacrificed. The EAF has very little forces to spare and must therefore deploy its mechs as efficient as possible.

You have been tasked to write a program that will lead the EAF to victory. Given a graph of shield towers, determine which channels must be destroyed to make all energy fields disappear, in such a way that the least number of EAF mechs are lost during the battle.

E.1 Input

- The first line of input consists of the positive integer number n, the number of test cases;
- Then, for each test case:
  - A line containing the positive integer number m ($2 < m \leq 100$), the number of towers in the base;
  - Per tower two lines:
    - A line containing three positive integer numbers i ($0 \leq i \leq m-1$), u_i ($1 \leq u_i \leq 50$) and c_i ($1 \leq c_i \leq m-1$): the (identification) number of the tower, the amount of mechs that the tower can take down and the number of channels respectively. The numbers are separated by a space;
    - A line containing c_i different positive integers, the towers that are connected to tower i. A tower cannot be connected to itself. The integers are separated by a space.

A given base will always be able to generate at least one energy field.

E.2 Output

For each test case, the output contains a line with one number: the minimum number of EAF mechs that will be lost during the battle to make all energy fields disappear.

E.3 Example

Input	Output
1 3 0 1 2 1 2 1 2 2 0 2 2 3 2 0 1	3

Table 6: Example input and output

F Blackjack

In the game of blackjack you play against the dealer. Blackjack is played with a regular deck of playing cards containing the cards 2, 3, 4, 5, 6, 7, 8, 9, 10, jack, queen, king and ace of various suits. (However the suits do not influence the game in any way.) The cards 2 through 10 have value 2 through 10. The jack, queen, and king cards have value 10. Each individual ace can count as either 1 or 11. We define the low value of a player as the lowest possible sum of the values of all the cards of a player, thus counting all aces as 1. The high value of a hand is the highest possible sum of the values that is still below 22, counting aces as 1 or 11 accordingly.

In this problem a basic blackjack variant is considered; there is no splitting, doubling, insurance or "blackjack".

The goal of the game is to maximize your (expected) profit or mimimize your (even more expected) losses. One hand of the game is played as follows:

- You place a bet of x euros;
- You get one card, then the dealer gets one card; You get a second card, and finally the dealer gets a second card;
- As long as your low value is below 22, you have two options:
  - "Hit" - take one extra card from the deck;
  - "Stand" - take no more cards.
- If your low value is above 21, you lose bet and the hand ends immediately.
- Now the dealer will play his cards;
- As long as the high value of the dealer is below 17, the dealer takes an extra card from the deck;
- If the low value of the cards of the dealer is above 21, you receive $2x$ euros, making a profit of x euros;
- If the high value of the dealer is greater than than your high value, you lose the bet;
- If the high value of the dealer equals your high value, the bet is returned to you;
- If the high value of the dealer is less than than your high value, you receive $2x$ euros, making a profit of x euros;
- At the end of the hand, all used cards are placed on a "discard pile" and will not return into play.

If at any time there are no cards left in the deck and the dealer has to get another card or you choose to get another card then the current hand is disregarded and the bet is returned to you.

In this problem, the sequence of cards left in the deck is known to you. You have to write a program to help yourself play optimally (ie. maximize profit).

F.1 Input

- The first line of input consists of the integer number n, the number of test cases;
- Then, for each test case:
  - One line with three integers $0 \leq c \leq 10,000$ and $0 \leq p \leq q \leq 100$: the number of cards left in the deck, the minimum bet and the maximum bet respectively;
  - $\lceil c/60 \rceil$ lines with a total of c characters ('2'-'9', 'T', 'J', 'Q', 'K', 'A'), representing the sequence of cards in the deck. The first character on the first line represents the first card that will come off the deck. Only the last line may contain less than 60 characters.

F.2 Output

For each test case, the output contains one line with one number: the highest possible profit.

F.3 Example

Input	Output
1 4 1 15 TTA8	15

Table 7: Example input and output

G Robintron

It is during the great war in the year 2240 that the Robinson family decides to leave the human empire, in search for more peaceful and quiet places. The Robinsons already have a planet in mind to which they wish to travel and want to get there as soon as possible. However, because all vehicles capable of spaceflight are being used for the war, the Robinson family has no way to escape to other planets by means of ordinary space flight. It is therefore that Joe Robinson, the father of the family, created the *Robintron*. The Robintron is a vehicle that is capable of travelling from one planet's surface to another by only using the forces of gravity.

It is the gravitational pull of a planet that keeps the Robintron on a planet's surface. This pull extends as far as the planet's *gravity well*, a circular area around the center of the planet, which is generated by the planet's mass. However, these gravity wells can also be used to assist the Robintron in leaving a planet by the way of *planet hopping*. Planet hopping can be performed when the Robintron is on the surface of a planet and comes within the area of another planet's gravity well. The Robintron uses the other planet's gravity well to gain enough momentum to escape into space, and to land on the other planet's surface. Leaving a planet and subsequently landing on another one takes no significant amount of time.

Planet hopping has some disadvantages. The Robintron must be within the gravity well of a planet other than the one it resides on, to be able to effectively use it for planet hopping. However, as the planets rotate around a star (which they all do in perfectly circular orbits), their position continuously changes according to their rotational speed. Thus, it can take up some time for the Robintron to enter the gravity well of a particular planet. Also, as soon as the Robintron chooses to enter the gravity well of a planet, it will move along with that planet around the star. Therefore, this could mean that another planet's gravity well may never come in reach of the Robintron.

Given a list of planets in a star system, a starting position and destination, write a program that determines how many days it will take (using the ceiling function, i.e. $ceiling(2.3) = 3$) for the Robintron to travel to its destination, given that it travels via the fastest route possible.

Because in most star systems all planets are positioned around the star in a disc, only two dimensions have to be taken into account. Also, a planet is assumed to be a single point in space, and the Robintron's coordinates are equal to those of the planet it resides on.

G.1 Input

- The first line of input consists of the integer number n, the number of test cases;
- Then, for each test case:
  - A line containing the positive integer number m ($0 < m < 1000$), the number of planets in the star system (excluding the star which always resides at position [0, 0]);
  - Per planet a line containing 4 numbers x ($-1e10 \leq x \leq 1e10$) and y ($-1e10 \leq y \leq 1e10$), the x-coordinate and y-coordinate of the planet relative to the star at the start of the Robintron's journey respectively; r ($0 < r \leq 1e5$) the radius of the planet's gravity well and s ($0 \leq s \leq 2\pi$) the speed, in radians per day, at which the planet rotates (counterclockwise) around the central star. The numbers are separated by a space.

The Robintron's journey always starts at the first listed planet and its destination is always the last listed planet, which is always reachable from the Robintron's starting position. The star (at position [0, 0]) is too hot for the Robintron and cannot be used for planet hopping.

G.2 Output

For each test case, the output consists of a line with one integer number: the number of days (rounded up, or 0 if the Robintron can reach its destination immediately) that the Robintron must travel to reach its destination.

G.3 Example

Input	Output
2 3 10.0 0.0 1.0 1.570796325 -13.0 0.0 3.1 3.14159265 17.0 0.0 4.1 0.0 5 10.0 0.0 1.1 1.0 12.0 0.0 1.1 2.0 13.0 0.0 1.1 1.0 0.0 11.0 1.1 3.14159265 14.0 0.0 1.1 1.0	3 7

Table 8: Example input and output

H Diamond Dealer

Mr. Chou is the flatworld diamond dealer. It is important that he knows the value of his (two dimensional) diamonds in order to be a succesful businessman. Mr. Chou is tired of calculating the values by hand and you have to write a program that makes the calculation for him.

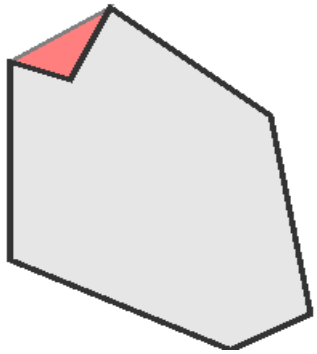


Figure 2: Example diamond

The value of a diamond is determined by smoothness of its surface. This depends on the amount of faces on the surface, more faces means a smoother surface. If there are dents (marked red in figure 2) in the surface of the diamond, the value of the diamond decreases. Counting the number of dents in the surface (a) and the number of faces on the surface that are not in dents (b), the value of the diamond is determined by the following formula: $v = -a \cdot p + b \cdot q$. When v is negative, the diamond has no value (ie. zero value).

H.1 Input

- The first line of input consists of the integer number n, the number of test cases;
- Then, for each test case:
  - One line containing:
    - The cost for a dent in the surface of a diamond ($0 \leq p \leq 100$);
    - The value of a face in the surface of a diamond ($0 \leq q \leq 100$);
    - The number of vertices ($3 \leq n \leq 30$) used to describe the shape of the diamond.
  - n lines containing one pair of integers ($-1000 \leq x_i, y_i \leq 1000$) describing the surface of the diamond $(x_0, y_0) - (x_1, y_1) - ... - (x_{n-1}, y_{n-1}) - (x_0, y_0)$ in clockwise order.

No combination of three vertices within one diamond will be linearly aligned.

H.2 Output

For each test case, the output contains one line with one number: the value of the diamond.

H.3 Example

Input	Output
1 10 5 7 0 10 8 4 10 -7 6 -9 -5 -4 -5 7 -2 6	15

Table 9: Example input and output