Stanford Local Programming Contest 2016

Gates B08/B12/B30

Saturday, October 1, 2016
2:00 pm to 6:00 pm

Read these guidelines carefully!

Contest website: `http://cs.stanford.edu/group/acm/slpclive`

Rules

1. You are expected to *adhere to the honor code*. You are still expected to conduct yourself according to the rules, even if you are not participating on site in the Gates building.
2. You may not collaborate in any way (verbally, electronically, in writing, using gestures, telepathically, etc.) with other contestants, students, or anyone else during the contest. The only exception is your teammates, only if you are competing as part of a team.
3. You may use any amount of written resource materials such as books, manuals, and program listings. You may *not* use any Internet resources, other than the following exceptions:
   - Official Stanford ACM-ICPC team notebook: `http://cs.stanford.edu/group/acm/SLPC/notebook.pdf`
   - Language references (e.g., the Java API documentation)
   - Digital versions of textbooks or course notes.

   You may *not*, however, electronically copy from machine-readable versions existing code or data. That is, all programs submitted must be *manually typed in their entirety during the contest*. No cutting and pasting of code is allowed!
4. You may bring and use your own laptop to write your code for this contest. To keep the computing environments as fair as possible across contestants, we ask that you restrict your use of software during the contest to text editors, IDEs, document readers, and a web browser for the sole purpose of connecting to the contest site (and perhaps a language reference). In other words, you may not use specialized tools such as Mathematica, Maple, or MATLAB, even if you have them installed on your machine.

Guidelines for submitted programs

1. All programs must be written in C++, Java, or Python. For judging, we will compile the programs in the following way:

- `.cpp`: using `g++ -g -O2 -std=gnu++11` (GCC 4.8.5)
- `.java`: using `javac -encoding UTF-8` (OpenJDK 1.7.0)
- `.py`: using `python2` (Python 2.7.6) or `python3` (Python 3.4.3)

All programs will be compiled and tested on a Leland `myth` machine. The `myth` machines are Intel Core2 Duo E6850 3.00 GHz machines with 8 GB RAM running Ubuntu 14.04. Compilation errors or other errors due to incompatibility between your code and the myth machines will result in a submission being counted incorrect.

2. Make sure your program terminates with a return value of zero. For example, in C++, your `int main()` function must `return 0;` **any non-zero return values may be interpreted by the automatic judge as a runtime error.**

3. **Java users**: Please place your `public static void main()` function in a public class with the same name as the base filename for the problem. For example, a Java solution for the `test` program should be submitted in the file `test.java` and should contain a `main()` in `public class test`.

4. All solutions must be submitted as a single file.

5. All programs should accept their input on **stdin** and produce their output on **stdout**. They should be batch programs in the sense that they do not require human input other than what is piped into stdin.

6. Be sure to follow the output format described in the problem exactly. We will be judging programs based on a `diff` of your output with the correct solution, so your program's output must match the judge output **exactly** for you to receive credit for a problem. As a note, each line of an output file must end in a newline character, and there should be no trailing whitespace at the ends of lines.

How will the contest work?

1. If you chose to work remotely from a home computer, we recommend that you test out your account on the online contest system by submitting a solution for the test problem shown on the next page. We will do our best to set up the contest host to accept test problem submissions Saturday afternoon until approximately 1:45 pm.

2. For those who choose to participate onsite, from 1:00 to 1:45 pm, you should select a computer (or find a place to plug in your laptop), set up your workspace and complete a test problem. Space in Gates is limited, and will be available on a first-come first-served basis.

3. At 2:00 pm, the problems will be posted on the live contest page in PDF format, all registered participants will be sent an e-mail that the problems have been posted, and we will distribute paper copies of the problems to contestants competing in Gates.

4. For every run, your solution will be compiled, tested, and accepted or rejected for one of the following reasons: *compile error*, *run-time error*, *time limit exceeded*, *incorrect output*, or *presentation error*. In order to be accepted, your solution must match the judge output exactly (according to `diff`) on a set of hidden judge test cases, which will be revealed after the contest.

   - Source code for which the compiler returns errors (warnings are ok) will be judged as *compile error*.
   - A program which returns any non-zero error code will be judged as *run-time error*.
   - A program which exceeds the time allowed for any particular problem will be judged as *time-limit exceeded* (see below).

- A program which fails a `diff -w -B` will be judged as *incorrect output*.
- A program which passes a `diff -w -B` but fails a diff (i.e., output matches only when ignoring whitespace and blank lines) will be judged as *presentation error*.
- A program which passes a `diff` and runs under the time constraints specified will be judged as **accepted**.

5. The time allowed for a run (consisting of multiple test cases) will be 1 second total for all test cases. The number of test cases in a run may vary depending upon the problem, so be sure to write algorithmically efficient code!

6. You can view the status of each of your runs on the live online contest site. Please allow a few minutes for your submissions to be judged. The site also provides a live scoreboard for you to watch the progress of the contest as it unfolds.

7. At 6:00 pm, the contest will end. No more submissions will be accepted. Contestants will be ranked by the number of solved problems. Ties will be broken based on total time, which is the sum of the times for correct solutions; the time for a correct solution is equal to the number of minutes elapsed since 2:00 pm plus 20 penalty minutes per rejected solution. No penalty minutes are charged for a problem unless a correct solution is submitted. After a correct submission for a problem is received, all subsequent submissions (correct or not) for that problem do not count towards the total time.

8. The results of this contest will be used in part to select team members for representing Stanford at the forthcoming ACM regional competition. Typically, Stanford sends five teams (of three members each) to the Pacific Northwest ACM-ICPC regional contest.

Helpful hints

1. **Make sure your programs compile and run properly on the `myth` machines**. These machines are accessible at Gates B08, or by remotely logging into `myth.stanford.edu` via `ssh`, for example. If you choose not to develop on the the `myth` machines, you are responsible for making sure that your code is portable. The `myth` machines will be the machines used for judging/testing of all programs.

2. **Read (or skim) through all of the problems at the beginning to find the ones that you can code quickly.** Finishing easy problems at the beginning of the contest is especially important as the time for each solved problem is measured from the beginning of the contest. Also, check the leaderboard frequently in order to see what problems other people have successfully solved in order to get an idea of which problems might be easy and which ones are likely hard.

3. If you wish to use an IDE (e.g., Visual Studio or Eclipse), please make sure that you know how to set this up yourself beforehand. We will not provide technical support related to setting up IDEs during the contest.

4. If you need a clarification on a problem or have any other questions, post an clarification request to the live contest page, or just come talk to us in **Gates 100**, or the **basement lobby area**.

The directions given here are originally based on those taken from Brian Cooper's 2001 Stanford Local Programming Contest problem set, and have been updated year after year to the best of our ability. The contest organizers would like to thank the problem authors of 2016, in alphabetical order, Jaehyun Park, Josh Alman, and Joshua Wang.

Problem A: Assembly Required

Princess Lucy broke her old reading lamp, and needs a new one. The castle orders a shipment of parts from the Slick Lamp Parts Company, which produces interchangable lamp pieces.

There are m types of lamp pieces, and the shipment contained multiple pieces of each type. Making a lamp requires exactly one piece of each type. The princess likes each piece with some value, and she likes a lamp as much as the sum of how much she likes each of the pieces.

You are part of the castle staff, which has gotten fed up with the princess lately. The staff needs to propose k distinct lamp combinations to the princess (two lamp combinations are considered distinct if they differ in at least one piece). They decide to propose the k combinations she will like the least. How much will the princess like the k combinations that the staff proposes?

Input

The first line of input contains a single integer T ($1 \leq T \leq 10$), the number of test cases. The first line of each test case contains two integers m ($1 \leq m \leq 100$), the number of lamp piece types and k ($1 \leq k \leq 100$), the number of lamps combinations to propose. The next m lines each describe the lamp parts of a type; they begin with n_i ($2 \leq n_i \leq 100$), the number of pieces of this type, followed by n_i integers $v_{i,1}, \ldots, v_{i,n_i}$ ($1 \leq v_{i,j} \leq 10{,}000$) which represent how much the princess likes each piece. It is guaranteed that k is no greater than the product of all n_i's.

Output

For each test case, output a single line containing k integers that represent how much the princess will like the proposed lamp combinations, in nondecreasing order.

Sample Input	**Sample Output**
2 2 2 2 1 2 2 1 3 3 10 4 1 5 3 10 3 2 3 3 5 1 3 4 6 6	2 3 4 5 5 6 6 7 7 7 7 7

Explanation

In the first case, there are four lamp pieces, two of each type. The worst possible lamp has value $1 + 1 = 2$, while the second worst possible lamp has value $2 + 1 = 3$.

Problem B: Bulbs

Greg has an $m \times n$ grid of Sweet Lightbulbs of Pure Coolness he would like to turn on. Initially, some of the bulbs are on and some are off. Greg can toggle some bulbs by shooting his laser at them. When he shoots his laser at a bulb, it toggles that bulb between on and off. But, it also toggles every bulb directly below it, and every bulb directly to the left of it. What is the smallest number of times that Greg needs to shoot his laser to turn all the bulbs on?

Input

The first line of input contains a single integer T ($1 \leq T \leq 10$), the number of test cases. Each test case starts with a line containing two space-separated integers m and n ($1 \leq m, n \leq 400$). The next m lines each consist of a string of length n of 1s and 0s. A 1 indicates a bulb which is on, and a 0 represents a bulb which is off.

Output

For each test case, output a single line containing the minimum number of times Greg has to shoot his laser to turn on all the bulbs.

Sample Input	**Sample Output**
2 3 4 0000 1110 1110 2 2 10 00	1 2

Explanation

In the first test case, shooting a laser at the top right bulb turns on all the bulbs which are off, and does not toggle any bulbs which are on.

In the second test case, shooting the top left and top right bulbs will do the job.

Problem C: Card Collecting

Lately, a variety of free-to-play collectible card games have become popular. These card games usually have some collection of n cards that the player wants to collect. The player is first given a starter pack of s cards, which is guaranteed to not contain any duplicates (they are all unique). The player can then start acquiring new card(s) in two different ways:

1. Trade any set of d cards with a card of the player's choice. This can be done very quickly, and for this problem, we will assume that it takes no time.
2. Play games for an hour, and win a pack of k cards with probability p_i, where i is the number of distinct cards that the player currently holds. The pack consists of cards that are independently and uniformly at random from the entire collection of n cards (so it may or may not have duplicates).

The advantage of having a larger collection, of course, is that a player has a higher chance of winning a pack, and hence earns random cards more quickly.

Stingy Larry plays card games of this type all the time. Larry just got a copy of such a game (and is just about to open his starter pack of s cards), and wants to complete a full collection. Assuming that Larry manages his collection optimally, what is the shortest expected time to complete the collection that he can achieve?

Input

The first line of the input contains a single integer T ($1 \leq T \leq 10$), the number of test cases. The first line of each test case contains four integers n ($1 \leq n \leq 100$), the total number of cards in the collection, s ($0 \leq s \leq n$), the starting number of cards, k ($1 \leq k \leq 10$), the number of cards in a pack, and d ($1 \leq d \leq 100$), the number of cards that must be traded for a new card. The next line of the test case then contains $n+1$ space-separated real numbers $p_0, p_1, \ldots, p_n$ ($0.01 \leq p_i \leq 1$), which represent the probabilities of Larry winning a pack after an hour of play. You may assume the p_i's are nondecreasing.

Output

For each test case, output the shortest expected time for Larry to complete the collection in hours. An answer is considered correct if its absolute or relative error is at most 10^{-7}.

Sample Input	**Sample Output**
2 1 0 1 1 0.25 1.0 10 2 5 5 0.01 0.1 0.2 0.3 0.4 0.5 0.5 0.5 0.5 0.5 1.0	4.000000000 10.185962459

Explanation

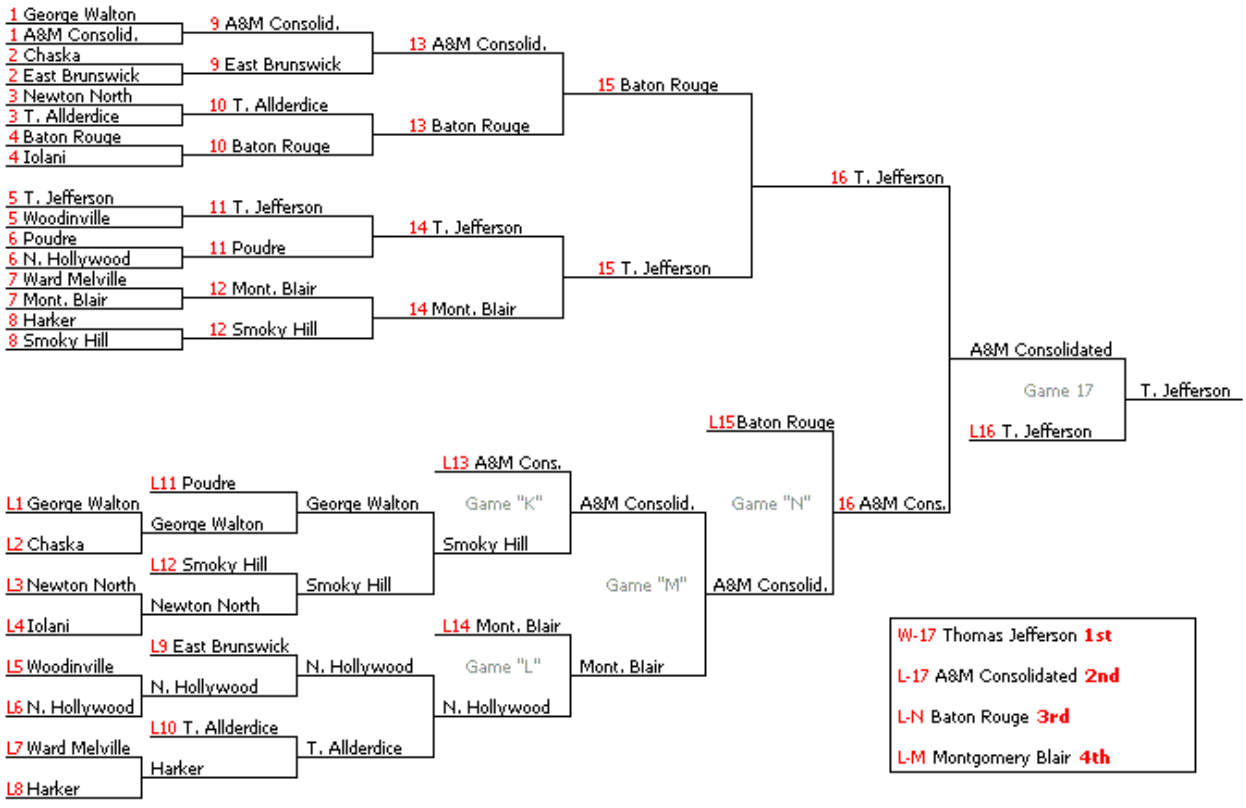
In the first test case, Larry will need to play an average of four hours to win the single card he needs to complete his collection.

Problem D: Double Elimination

Your friend J loves fighting games and plays them night and day. Everytime you try to play with J, he always wins in a crushing victory. You eventually suggest that J enter a fighting tournament, where he can play with people closer to his level. J decides to enter the Street League Pixel Championship, which is a bracketed double-elimination tournament.

Bracketed double-elimination tournaments work as follows. There are two brackets: the winners bracket and the losers bracket. Everyone begins in the winners bracket. Each round, players left in the winners bracket are paired and play matches. Winners remain in the winners bracket and losers drop down to the losers bracket.

Rounds of the losers bracket consist of a minor stage followed by a major stage. Suppose that at the beginning of a round, the losers bracket contains x players. In the minor stage, the x players in the losers bracket are paired and play matches. The $x/2$ winners from these matches remain in the losers bracket, and the other $x/2$ players are eliminated from the tournament. When $x/2$ players drop down from the winners bracket into the losers bracket, the major stage begins, where the new pool of x players are paired and play matches. The $x/2$ winners of this stage remain in the losers bracket, and the other $x/2$ players are eliminated from the tournament. This continues until there is only a single player in each bracket. These two players then play a match. If the player from the winners bracket wins, they win the tournament. Otherwise, the player from the player from the winners bracket drops down to the losers bracket and they play a final match (grand finals). The winner of that match wins the tournament.



Since you are a good friend, you watch all of J's games. J wins w games in the winner's bracket and ℓ games in the loser's bracket. What is J's final rank in the tournament? Participants are ranked by when they are eliminated from the tournament. Everyone eliminated at the same time is considered to be in a tie for the best possible rank they could all be. The winner of the tournament is rank one.

Input

The first line of input contains a single integer T ($1 \leq T \leq 10{,}000$), the number of test cases. The next T lines of input each contain three integers k ($0 \leq k \leq 30$), w, ℓ, representing a valid double-elimination tournament with 2^k competitors where J wins w games in the winner's bracket and ℓ games in the loser's bracket.

Output

For each double-elimination tournament, output a single line with a single integer giving J's rank.

Sample Input	**Sample Output**
3 2 1 3 2 3 0 3 0 0	1 1 7

Explanation

In the first tournament, there are four competitors. J wins his first game in the winners bracket, after which there are two players in the winners bracket and two players in the losers bracket. J loses his next game in the winners bracket, dropping down to the losers bracket. There is now one player in the winners bracket and two players in the losers bracket. J proceeds to beat the other player in the losers bracket to make it to the finals, where he defeats the winners bracket player twice to take the tournament.

In the second tournament, there are four competitors. J wins two games in the winners bracket to make it to the finals, where he wins to take the tournament.

In the third tournament, there are eight competitors. Believe or not, J drops out immediately, as does one other player, making the two tied for rank seven.

Problem E: Election of Evil

Dylan is a corrupt politician trying to steal an election. He has already used a mind-control technique to enslave some set U of government representatives. However, the representatives who will be choosing the winner of the election is a different set V. Dylan is hoping that he does not need to use his mind-control device again, so he is wondering which representatives from V can be convinced to vote for him by representatives from U.

Luckily, representatives can be persuasive people. You have a list of pairs (A, B) of represenatives, which indicate that A can convice B to vote for Dylan. These can work in chains; for instance, if Dylan has mind-controlled A, A can convince B, and B can convince C, then A can effectively convince C as well.

Input

The first line contains a single integer T ($1 \leq T \leq 10$), the number of test cases. The first line of each test case contains three space-separated integers, u, v, and m ($1 \leq u, v, m \leq 10{,}000$). The second line contains a space-separated list of the u names of representatives in U. The third line contains a space-separated list of the v names of representatives from V. Each of the next m lines contains a pair of the form `A B`, where `A` and `B` are names of two representatives such that `A` can convince `B` to vote for Dylan. Names are strings of length between 1 and 10 that only consists of lowercase letters (`a` to `z`).

Output

For each test case, output a space-separated list of the names of representatives from T who can be convinced to vote for Dylan via a chain from S, in alphabetical order.

Sample Input	**Sample Output**
2 1 1 1 alice bob alice bob 5 5 5 adam bob joe jill peter rob peter nicole eve saul harry ron eve adam joe chris jill jack jack saul	bob peter saul

Explanation

In the second test case, Jill can convince Saul via Jack, and Peter was already mind-controlled.

Problem F: Flight Plan

You work for SLPC Airlines, a small airline company that has just found a critical bug in its flight planner. The bug causes flights to take paths that keep either latitude or longitude constant. For example, to route a plane from $(0, 0)$ to $(10, 10)$, the flight planner would send the plane from $(0, 0)$ to $(0, 10)$ in a path keeping the latitude constant, and then from $(0, 10)$ to $(10, 10)$ in a path keeping the longitude constant.

You want to know how much the company has been losing from this bug. As a first step, you have the latitude/longitude coordinates of various cities and want to compute the shortest possible flight distance between them and the bugged flight distance between them.

Assume that the Earth is a perfect sphere with radius equal to 6,371 km, and that SLPC aircraft fly at a negligible height above the surface of the Earth.

Latitude and Longitude Reminder

The latitude of a point on the Earth's surface is the angle between the plane of the equator and the line going through the center of the Earth and that point. Latitudes range from 90°S (the South Pole) to 90°N (the North Pole).

The longitude of a point on the Earth's surface is the angle between a plane containing the Prime Meridian and a plane containing the North Pole, the point in question, and the South Pole (a meridian is half of a great circle from the North Pole to the South Pole; the Prime Meridian is a meridian chosen to have longitude 0). Latitudes range from 180°W to 180°E.

Input

The first line of input contains a single integer T ($1 \leq T \leq 10{,}000$), the number of test cases. Each test case is a single line with four real numbers a_1, b_1, a_2, b_2 ($|a_1|, |a_2| \leq 90$, and $|b_1|, |b_2| \leq 180$). These numbers form a pair of latitude/longitude coordinates (a_1, b_1) and (a_2, b_2), in degrees. Positive latitude coordinates represent points north of the equator; negative, south. Positive longitude coordinates represent points east of the prime meridian; negative, west.

Output

For each test case, output a single line giving the shortest possible flight distance between them followed by the bugged flight distance between them. An answer is considered correct if its absolute or relative error is at most 10^{-7}.

Sample Input	**Sample Output**
2 -90 0 90 0 0 0 10 10	20015.08679602 20015.08679602 1568.52055680 2223.89853289

Explanation

In the first case, the bugged flight planner happens to pick the best route between the two points; both distances are exactly half of a great circle.

In the second test case, the bugged flight planner does happen not pick the best route, so the answers differ.

Problem G: Ground Defense

You are a denizen of Linetopia, whose n major cities happen to be equally spaced along an east-west line. In fact, they are often numbered in order from 1 to n, where 1 is the westmost city and n is the eastmost city. Linetopia was a lovely place to live until forces from neighboring Trapez invaded. As part of Linetopia's Shielding Lives and Protecting Citizens initiative, you have been called upon to process information about Trapezoid troop movements so we can determine which cities have been hardest hit and know where to send reinforcements.

Linetopia intelligence has discovered that the Trapezoid forces are attacking in the following pattern. They are sending massive aircraft to drop troops on Linetopia cities. Each aircraft starts at some city i, dropping s soldiers. The aircraft then proceeds to fly either east or west. Each time it flies over another city, it drops a more soldiers than it dropped on the previous city it passed. After performing d drops, the aircraft returns to Trapez to resupply.

You will be receiving intel updates that inform you of the specs of each Trapezoid aircraft passing over Linetopia. You want to answer queries that ask how many Trapezoid troops have been dropped on a particular city. Are you up to the task?

Input

The first line of input contains a single integer T ($1 \leq T \leq 10$), the number of test cases. The first line of each test case contains two integers: m ($1 \leq m \leq 10{,}000$), the number of updates and queries and n ($1 \leq n \leq 500{,}000$), the number of cities in Linetopia.

The next m lines of input are either updates or queries. Update lines begin with a capital `U`, then contain either a capital `E` (east) or `W` (west) to indicate direction, and then contain four integers i ($1 \leq i \leq n$), s ($1 \leq s \leq 10{,}000$), a ($0 \leq a \leq 10{,}000$), and d ($1 \leq d \leq n$). These integers signify the starting city, the starting number of soldiers, the increase in soldiers per city, and the number of drops, respectively. You can assume d never results in an aircraft flying to the west of city 1 or to the east of city n.

Query lines begin with a capital `Q`, and then contain a single integer i ($1 \leq i \leq n$) indicating the city being queried.

Output

For each query in the input, output a single line containing the number of Trapezoid troops dropped in that city.

Sample Input	Sample Output
1 8 3 U E 1 5 2 3 Q 1 Q 2 Q 3 U W 3 10 10 2 Q 1 Q 2 Q 3	5 7 9 5 27 19

Explanation

Two aircrafts fly over Linetopia. The first starts at city 1 and heads east. It drops 5 soldiers on city 1, 7 soldiers on city 2, and 9 soldiers on city 3. The second starts at city 3 and flies west. It drops 10 soldiers on city 3 and 20 soldiers on city 2.

Problem H: Hunter's Apprentice

When you were five years old, you watched in horror as a spiked devil murdered your parents. You would have died too, except you were saved by Rose, a passing demon hunter. She ended up adopting you and training you as her apprentice.

Rose's current quarry is a clock devil which has been wreaking havoc on the otherwise quiet and unassuming town of Innsmouth. It comes out each night to damage goods, deface signs, and kill anyone foolish enough to wander around too late. The clock devil has offed the last demon hunter after it; due to its time-warping powers, it is incredibly agile and fares well in straight-up fights.

The two of you have spent weeks searching through dusty tomes for a way to defeat this evil. Eventually, you stumbled upon a relevant passage. It detailed how a priest managed to ensnare a clock devil by constructing a trap from silver, lavender, pewter, and clockwork. The finished trap contained several pieces, which must be placed one-by-one in the shape of a particular polygon, in counter-clockwise order. The book stated that the counter-clockwise order was important to counter the clock devil's ability to speed its own time up, and that a clockwise order would only serve to enhance its speed.

It was your responsibility to build and deploy the trap, while Rose prepared for the ensuing fight. You carefully reconstruct each piece as well as you can from the book. Unfortunately, things did not go as planned that night. Before you can finish preparing the trap, the clock devil finds the two of you. Rose tries to fight the devil, but is losing quickly. However, she is buying you the time to finish the trap. You quickly walk around them in the shape of the polygon, placing each piece in the correct position. You hurriedly activate the trap as Rose is knocked out. Just then, you remember the book's warning. What should you do next?

Input

The first line of input contains a single integer T ($1 \leq T \leq 100$), the number of test cases. The first line of each test case contains a single integer n ($3 \leq n \leq 20$), the number of pieces in the trap. Each of the next n lines contains two integers x_i and y_i ($|x_i|, |y_i| \leq 100$), denoting the x and y coordinates of where the ith piece was placed. It is guaranteed that the polygon is simple; edges only intersect at vertices, exactly two edges meet at each vertex, and all vertices are distinct.

Output

For each test case, output a single line containing either `fight` if the trap was made correctly or `run` if the trap was made incorrectly.

Sample Input	Sample Output
2 3 0 0 1 0 0 1 3 0 0 0 1 1 0	fight run

Explanation

In the first case, you went around the polygon in the correct direction, so it is safe to fight the clock devil and try to save Rose.

In the second case, you messed up, and it is time to start running. Sorry Rose!

Problem I: Ingenious Lottery Tickets

Your friend Superstitious Stanley is always getting himself into trouble. This time, in his Super Lotto Pick and Choose plan, he wants to get rich quick by choosing the right numbers to win the lottery. In this lottery, entries consist of six distinct integers from 1 to 49, which are written in increasing order. Stanley has compiled a list of winning entries from the last n days, and is going to use it to pick his winning numbers.

In particular, Stanley will choose the six numbers that appeared the most often. When Stanley is breaking ties, he prefers smaller numbers, except that he prefers seven to every other number. What is Stanley's entry?

Input

The first line of input contains a single integer T ($1 \leq T \leq 100$), the number of test cases. The first line of each test case contains a single integer n ($1 \leq n \leq 1{,}000$), the number of winning entries that Stanley compiled. The next n lines each contain a lottery entry as described above.

Output

For each test case, output a single line containing Stanley's entry.

Sample Input	**Sample Output**
2 3 1 2 3 4 5 6 4 5 6 7 8 9 7 8 9 10 11 12 3 1 2 3 4 5 6 4 5 6 7 8 9 1 2 3 7 8 9	4 5 6 7 8 9 1 2 3 4 5 7

Explanation

In the first test case, the numbers 4 through 9 appear twice each, while all other numbers appear at most one time.

In the second test case, all numbers 1 through 9 appear twice each. The tiebreaking rule means Stanley prioritizes picking 7 and then the five smallest numbers.

Problem J: Jurisdiction Disenchantment

The Super League of Paragons and Champions (SLPC) has been monitoring a plot by a corrupt politician to steal an election. In the past week, the politican has used a mind-control technique to enslave the n representatives responsible for choosing the election's winner. Luckily, the SLPC has managed to recruit you and hence has access to your power to break mind-control. You are able to break mind-control in an axis-aligned rectangle. Unfortunately, your power comes at a steep cost; you get a headache the next day proportional to the size of the rectangle. You do not even want to risk or think about what would happen if you tried to use your power multiple times in one day.

You have done your research and you know the position that each representative will be standing when the votes are about to be cast. You need to free enough representatives to prevent the politician from having a majority (strictly more than one-half) vote. What is the area of the smallest axis-aligned rectangle that you can affect to do this?

Input

The first line of input contains a single integer T ($1 \leq T \leq 10$), the number of test cases. The first line of each test case contains a single integer n ($1 \leq n \leq 299$, n is odd), the number of representatives. Each of the next n lines of input contains two integers, denoting the x and y coordinates of a representative. It is guaranteed that all coordinates are between $-10{,}000$ and $+10{,}000$.

Output

For each test case, output a single line containing the area of the smallest axis-aligned rectangle containing more than $n/2$ of the representatives.

Sample Input	**Sample Output**
2 1 1 1 3 0 0 1 4 3 2	0 4

Explanation

In the first case, a rectangle containing a single point has an area of 0.

In the second test case, the rectangle needs to include at least two points. There are two smallest possible rectangles; one includes $(0, 0)$ and $(1, 4)$ and the other includes $(1, 4)$ and $(3, 2)$. In either case, the area is 4.