VCU School of Engineering
2016 Computer Science Programming Contest

Welcome to the 2016 Programming Contest. Before you start the contest, please be aware of the following:

1. There are ten (10) problems in the packet, using letters A-J. These problems are NOT sorted by difficulty. When a team's solution is judged correct, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	Nearest Neighbors	Blue
B	Secret Channels	Green
C	Within Speed Limit	Orange
D	Rasterizing Triangles	Pink
E	Protein Similarity	Purple
F	Polish Notation	Silver
G	On a Child's Whim	Yellow
H	Graph Drawing	Gold
I	Shooting in the Dark	Black
J	Stop the Rabbits	White

2. All solutions must read from standard input (System.in) and write to standard output (System.out).

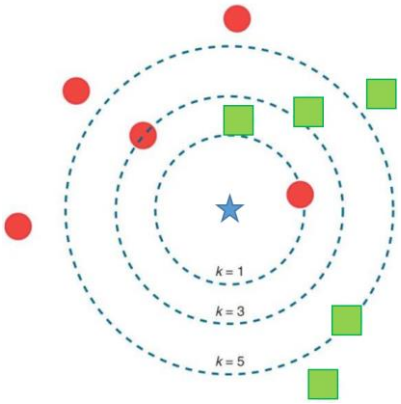
3. Solutions for problems submitted for judging are called runs. Each run will be judged. Runs for each particular problem will be judged in the order in which they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for B followed by a run for C, but receive the response for C first. DO NOT request clarifications on when a response will be returned. If you have not received a response for a run within 10 minutes of submitting it, **you may ask the lab monitor to send a runner to the judge to determine the cause of the delay. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.**

The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
Yes	Your submission has been judged correct.
Wrong Answer	Your submission generated output that is not correct.
Output Format Error	Your submission's output is not in the correct format, is misspelled, or did not produce all of the required output.
Excessive Output	Your submission generated output in addition to what is required.
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	You submission did not solve the judges' test data within **2 minutes**.

4. A team's **score** is based on the **number of problems they solve and penalty points**. The penalty points reflect the time required to solve a problem correctly and the number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved. Teams are ranked by the number of problems solved. Ties are resolved in favor of the team with the fewest penalty points.
5. This problem set contains sample input and output for each problem. However, you may be assured that the judges will test your submission against several other more complex datasets, which will not be revealed. Before submitting your run you should design other input sets to fully test your program. Should you receive an incorrect judgment, you are advised to consider what other datasets you could design to further evaluate your program.
6. In the event that you think a problem statement is ambiguous, you may request a clarification. Read the problem carefully before requesting a clarification. If the judges believe that the problem statement is sufficiently clear, you will receive the response, "The problem statement is sufficient, no clarification is necessary." If you receive this response, you should read the problem description more carefully. If you still think there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found. If the problem statement is ambiguous in specifying the correct output for a particular input, please include that input data in the clarification request.
7. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.
8. Good luck and HAVE FUN!!

Problem A
Nearest Neighbors



Classification is a machine learning task that predicts the class membership of unknown objects based on the information from other known objects. Classes of objects are denoted by positive integers – e.g. if objects are animals, class 1 can represent mammals, class 2 reptiles, class 3 birds, etc. The K-Nearest Neighbors (KNN) classifier predicts the class membership of an object by a majority vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer). If k = 1, then the object is simply assigned to the class of that single nearest neighbor.

Input

Input consists of a number of X known objects with Y dimensions (e.g. an object with two numerical attributes, such as height and weight, resides in two-dimensional space, Y=2) plus their class membership, Z unknown objects to be classified, and the K value (K < X).
First line of input contains: positive integers X, Y, Z, K
X following lines: Y real numeric values + class value (known objects)
Z following lines: Y real numeric values (unknown objects)

Output

Output should consist of Z lines. Each line is the class prediction for the unknown objects from the input. Euclidean distance between objects is employed. In case of a tie in the number of votes, the class with the lowest numerical value in the label is chosen.

Sample Input

```
12 4 3 1
5.1 3.5 1.4 0.2 1
4.9 3.0 1.4 0.2 1
4.7 3.2 1.3 0.2 1
4.6 3.1 1.5 0.2 1
5.0 3.6 1.4 0.2 1
7.0 3.2 4.7 1.4 2
6.4 3.2 4.5 1.5 2
6.9 3.1 4.9 1.5 2
5.5 2.3 4.0 1.3 2
6.3 3.3 6.0 2.5 3
5.8 2.7 5.1 1.9 3
7.1 3.0 5.9 2.1 3
5.0 3.3 1.4 0.2
5.7 2.8 4.1 1.3
5.9 3.0 5.1 1.8
```

Sample Output

```
1
2
3
```

Problem B
Secret Channels

A company uses public Internet to carry its phone service. The voice data will be encrypted before sending to Internet. The encryption algorithm is as follows: For each four-digit decimal integer, (1) Add 5 to each digit, divide the sum by 10 and use the remainder to replace the digit, (2) Swap 1^{st}-digit with 4^{th}-digit, (3) Swap 2^{nd}-digit with 3^{rd}-digit.

Input
A series of lines. Each line is a positive four-digit decimal integer.

Output
In each line, print the encrypted four-digit decimal integer.

Sample Input
```
0123
5890
9999
```

Sample Output
```
8765
5430
4444
```

Problem C
Within Speed Limit

Country C's inter-city transportation relies on road network to connect towns and cities. Each highway may have different speed limits, between 40mph to 70mph. We use a formatted string to represent a highway. For example, "50 1 40 2 60 3" represents a highway that connects town1, town2, and town3 together. The speed limit on the highway is 50 miles/hour. The section between town1 and town2 is 40 miles and town2 to town 3 is 60 miles. Note that all highways are bi-directional and there can be multiple highways to connect two places.

Jasper is planning to travel from townX to townY. Given the road map represented by the strings above, can you estimate the least amount of time it takes for Jasper to drive from townX to townY? Assume that Jeff is a law-abiding citizen and he will not drive over the speed limit, ever.

Input

Input consists of multiple lines of strings.

The first string is of format "nTowns townX townY", which means the total number of towns, the starting town number, and the destination town number. The destination will be reachable from the starting point. Towns are numbered by consecutive nonnegative integers, starting from 0. All following lines each represents a highway, in the string format outlined in the problem description above.

Output

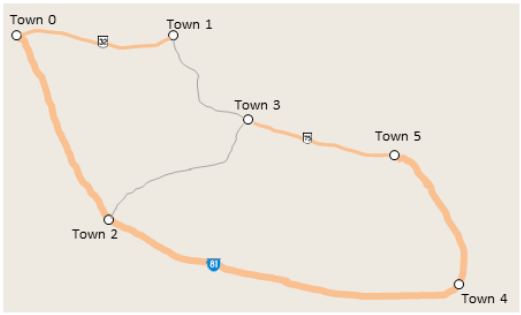
The least amount of time that Jasper needs to drive from the source to the destination, in hours (e.g 3.5 means 3 hours, 30 minutes).

Sample Input

```
6 0 5
60 0 60 2 120 4 60 5
50 0 50 1
30 1 50 3
40 2 40 3
50 3 75 5
```

Sample Output

```
3.5
```



Problem D
Rasterizing Triangles

In computer graphics, a common task is to rasterize an image, converting geometrical shapes into a 2D array of pixels to be rendered on the screen. In this problem, we will rasterize randomly selected pixels of an image consisting only of a single triangle. In order to do this correctly, we must determine whether each pixel is inside the triangle and should get the triangle's color, or if it's outside the triangle and should receive the background color, whatever that is.

Hint: Don't assume that the triangle's vertices are sorted in any way in the input

Input

Input consists of a triangle (described by three 2D vertices) and the coordinates of the pixel. For simplicity, each triangle and pixel coordinate is positive and a multiple of 0.5 (i.e. 0.0, 0.5, 1.0, 1.5, etc.).

First line: six, space delimited, coordinates describing the 3 vertices of the triangle (i.e. x1 y1 x2 y2 x3 y3)
Second line: the number of pixels to evaluate
Each subsequent line: a pixel vertex, as two, space delimited, coordinates (px py)

Output

Output is 0 if the pixel is outside of the triangle and 1 if it's inside

Sample Input

```
1.0 1.0 2.0 2.0 3.0 1.0
2
2.0 1.5
0.5 1.5
```

Sample Output

```
1
0
```

Problem E
Protein Similarity

Proteins are sequences of amino acids that are represented by capital letters (each letter corresponds to a unique amino acid). There are many (millions) of different proteins and it is useful to find out how similar their sequences are. The simplest way to quantify this similarity between two proteins is to compute number of amino acids (letters) that are at the same position in both sequences and that are the same and divide this number by the length of the shorter protein.

Input

Input consists of two protein sequence that have the same length $n < 500$
First line: Sequence of the first protein
Second line: Sequence of the second protein

Output

Output should consist of 1 line that gives the value of similarity expressed as the number of the same amino acids / n (without spaces).

Sample Input

```
MGINTRELFLNFTIVLITVILMWLLVRSYQY
MERSTQELFINFTVVLITVLLMWLLVRSYQY
```

Sample Output

```
24/31
```

Problem F
Polish Notation

Computer scientists have a strange way of looking at arithmetic expressions. What a regular person sees as 3+4*5 looks like + 3 * 4 5 to a computer scientist. The former is called the infix notation, while the latter is called the prefix notation, or the Polish notation. In the Polish notation, the operator (e.g. +) comes first, followed by the left-hand side sub-expression, followed by a right-hand side subexpression. Each of the subexpressions again is in the prefix notation. One benefit is that no brackets are needed, they can be inferred: + 3 * 4 5 can be unambiguously understood as + 3 (* 4 5), that is, as 3+(4*5).

You task is to write a converter from prefix notation to infix notation for expressions that involve numbers in the range 0-9 (that is, every number is a single digit, there won't be numbers such as 123), as well as two binary operators: + and *. The converter should preserve the left-right order, that is, + 3 4 is translated to 3+4 and not to 4+3. To simplify things, every binary operation should be enclosed in brackets: instead of 3+4*5 you should print (3+(4*5)).

Input

A string of length up to 100 characters representing an expression in prefix notation. The string will contain only the following characters: 0123456789+* and each character will be separated from the next by a single white space.

Output

A string representing the expression in infix notation. Do not use any white spaces.

Sample Input

```
+ * 3 + + 4 5 6 7
```

Sample Output

```
((3*((4+5)+6))+7)
```

Problem G
On a Child's Whim

It is well-documented that children can be fussy, almost to an illogical extent. For example, given two pieces of candy x and y, a child might absolutely *insist* on having candy x and not candy y, even if x and y are identical in all respects. Such are the trials and tribulations each parent must deal with. For this problem, you are to imagine that you are in a bulk candy store with n bins of candy arranged in a line in front of you (i.e. numbered 1 through n). For each pair of bins, your child has two predetermined preferences which would be acceptable. For example, given the pair of bin indices (1,2), your child may be happy if either both candies 1 and 2 are purchased, or both candies 1 and 2 are not purchased (i.e. purchasing 1 but not 2 will lead to a temper tantrum). Your job is to decide, given for each pair of consecutive bin indices (i,i+1) a pair of such preferences, which bins to take candies from so as to maximize the number of bin pairs (i,i+1) for which your child's preferences are met.

Input

Input consists of the following:

- An integer m between 1 and 1024 denoting the number of instances you need to solve. For each such instance, you will read the following below:
- An integer n between 1 and 1024 corresponding to the number of bins.
- For each bin pair (i,i+1) for 1 <= i <= n-1, bits a,b,c,d from the set {0,1} as follows:

  a b c d

  where e.g. if a=1, b=0, c=1, d=1, then due to the values of a and b it is acceptable for you to purchase candy i but not i+1, OR due to the values of c and d you may purchase both candies i and i+1. Note that whether your child wants candy i or not can change depending on whether one is considering the pair (i-1,i) or (i,i+1). Each such pair of preferences is given on a separate line.

Output

For each instance solved, output on a new line a single integer denoting the maximum number of possible preferences which can be satisfied simultaneously.

Sample Input 1

```
1
5
0 0 0 1
0 0 1 0
0 0 0 1
1 1 1 0
```

Sample Output 1

```
4
```

Sample Input 2

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

Sample Output 2

```
4
3
```

Problem H
Graph Drawing

A drawing of a graph or network diagram is a pictorial representation of the vertices and edges of the graph. The vertices are (often) represented as disks and the edges are represented as line segments. In this problem, you are asked to compute the area of the union of all the disks in a graph drawing. Knowing this area is important to create a well-balanced and beautiful drawing.

Input

The input contains multiple datasets. The first line in each dataset is one integer n indicates the number of the disks (or vertices). ($1 \leq n \leq 200$). Then follows n lines: every line has three integers $X_i\ Y_i\ R_i$ indicating the coordinates of the center of the disk, and the radius of the disk ($|X_i|. |Y_i| \leq 5000, R_i \leq 10$). The end of the input is marked by -1.

Output

For each dataset, output the area of the union of all the disks in that dataset, with 1 digit after decimal point.

Sample Input

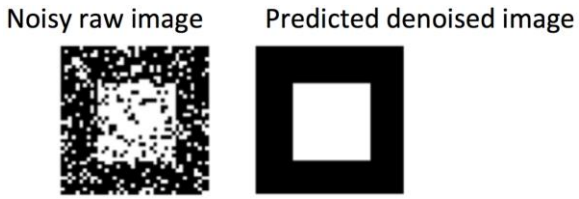
```
3
0 0 1
0 0 1
100 100 1
2
0 0 1
1 1 1
-1
```

Sample Output

```
6.3
5.7
```

Problem I
Shooting in the Dark

When a camera records a black-and-white image in difficult conditions, for example when it is quite dark, the raw image may have a lot of noise, as in the example below.



How to recover the denoised image (on the right) from a noisy raw image (left)? That is, how to predict assignments of 0 (black) or 1 (white) to each pixel? We can assume that recorded raw pixels are correct more often than not. Also, we can make an assumption that neighboring pixels should often have the same value: for example, if we have one white pixel surrounded by black pixels, probably that is the effect of noise, and the pixel should be black.

These two assumptions can be turned into a method for predicting the best 0/1 (black/white) assignment P(x) for each pixel x, by designing a scheme that assigns a penalty to every possible predicted denoised image, and picks the image with lowest penalty:

- If two neighboring pixels x and y have different predictions of their color, that is, if $P(x) \neq P(y)$, there will be a mismatch penalty $M(\{x,y\})$. The value of $M(\{x,y\})$ may be different for each unordered pair of neighboring pixels {x,y}. For pairs of pixels that are not neighbors, penalty is not defined: we do not care if they predict the same color or different colors.
- For any pixel x, predicting value 0 (black) will lead to penalty B(x), and predicting value 1 (white) will lead to penalty W(x). That is, the penalty is B(x) if we predict P(x)=0 and W(x) if we predict P(x)=1 in the denoised image. For example, if the raw recorder color for pixel x was 0 (black), we may set up penalties for that pixel as B(x)=1 and W(x)=20, that is, a much higher penalty for predicting x to be 1 (white). Penalties for different pixels may differ.

Now, assume you make a prediction of the denoised image by assigning P(x) for every pixel x. You can judge how good that prediction is by calculating the aggregated penalty M+B+W based on the predictions P(x). That is, you would calculate the sum of all penalties $M(\{x,y\})$ over all neighboring pixels x, y, plus sum of penalties B(x)+W(x) over all pixels x. The best denoised image, the one that will be returned to the user, is the one that has the lowest aggregated penalty M+B+W.

You will be given penalty values $M(\{x,y\})$ for all neighboring pairs of pixels x, y in the picture (for most pairs {x,y} there will be no penalty specified on input; you should conclude that these pixels are not neighbors), and penalty values B(x) and W(x) for all pixels x in the picture. Your task is to make, for every pixel x, an assignment P(x). The set of assignments over all pixels should lead to the lowest possible aggregated penalty M+B+W. All individual penalties $M(\{x,y\})$, B(x) and W(x) are non-negative integers not exceeding 100.

Input

The first line specifies the number of pixels in the image, 2<=n<=200.
Second line specifies the number of pairs of neighboring pixels, 1<=m<=10000.
The next n lines describe the pixel penalties B and W. Each line has three numbers separated by spaces: pixel number x (in the range 1,…,n), followed by B(x), followed by W(x).
The following m lines specify the pairs of neighboring pixels, and the associated mismatch penalties. Each line has three numbers separated by spaces: pixel number x, then pixel number y, and then the mismatch penalty value M({x,y}).

Output

Two numbers characterizing the best denoised image: number of pixels that have assignment P(x)=0 (i.e., black), followed by the total penalty M+B+W of the best denoised image.

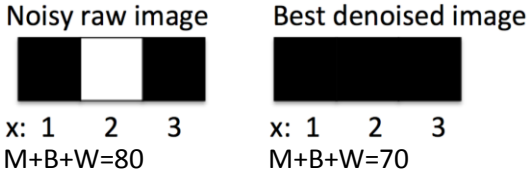
Sample Input

```
3
2
1 10 50
2 50 0
3 10 50
1 2 30
2 3 30
```

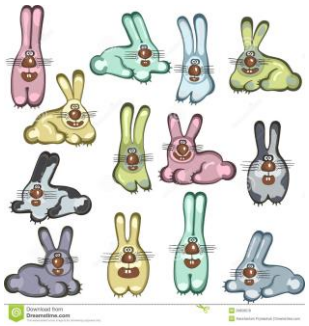
Sample Output

```
3 70
```

Raw and best denoised image corresponding to sample input/output



Problem J
Stop the Rabbits



One of New Zealand's early colonists brought with him a few rabbits, which soon escaped into the open country. Kiwis (as New Zealanders call themselves) didn't count on two facts: first, that there are no natural enemies of rabbits in New Zealand and second, that a female rabbit can give birth to about 30 baby rabbits each spring. These mature quickly, and within a year can have litters of their own. And so, a few years later, hordes of rabbits were overrunning the farms and destroying crops. The entire population of Kiwis was drafted into a war of extermination. (If you were the gardener, you knew the havoc that a too many rabbits can wreak). In the XXI century, genetic engineering was tried to stop the rabbits – the effect was catastrophic, the rabbits mutated and now they never die of natural causes, never lose their reproductive capabilities, and they started killing other small animals! The Kiwis eventually found a way to design a rabbit-killing AI, but it is not perfect – it can only be used in winter, because it overheats, and it only kills a fraction of rabbits.

Now the Kiwis are trying to figure out if the AI will be enough to keep the killer rabbits in check! They enlisted you to implement a model of rabbit population growth. It operates on a summer-to-summer basis. It assumes there are E female rabbits and E male rabbits in the summer of year one. It assumes every winter, the AI will kill a percentage p of all rabbits, males and females in equal proportion (if after this slaughter you end up with 30.7 males and 32.8 females, round these down). It assumes that every surviving female will give birth to R baby rabbits in spring, and if R is odd, the odd baby rabbit will be a female. These rabbits will reach reproduction age by next spring (if they survive the AI winter) and will be ready to reproduce.

Ultimately, the Kiwis would like to be able to enter E, p, and R parameters into the model, and get an answer of how many rabbits are expected to be alive in year Y from the start of the model. For example, if there were 100 female and 100 male rabbits in the summer of year 1, the query for year one will return 200, no matter what p and R values are. If the query is for year 2, and the parameters are p=50, R=9, there will be 50 females and 50 males surviving into spring of year 2, and each of those 50 females will bear 5 female baby rabbits and 4 male baby rabbits that spring. So by summer of year 2, there will be 550 rabbits: 300 females and 250 males. In spring of year three, 150 surviving female rabbits will again bear 5 female and 4 male baby rabbits to add to the population.

Input

Input consists of a number of *n* ($n \le 1000$) independent problems to be solved.
Each following line contains four positive integers: $E \le 1000$, $p \le 100$, $R \le 50$ and $Y \le 10$.

Output

Output should consist of *n* lines.
Each line is a solution to a single problem from the input. Each line should be a single positive integer: the number of rabbits alive in the summer of year Y from the start of the model. The number may be large, but will not exceed 1,000,000,000.

Sample Input

```
3
100 50 9 1
100 50 9 2
100 50 9 3
```

Sample Output

```
200
550
1625
```