JOI 2011-2012 Qualification contest

Authors: JCIOI (Japanese Committee for the IOI)

Translators: Jasmine Chua, France-ioi

License: Creative Commons Attributions-ShareAlike 4.0 International License (CC BY-SA 4.0)

**Lunch**

**QUESTION**

One of the most popular dishes in JOI's pasta restaurant is the lunch set with pasta of the day and fresh juice. The set allows you a choice of 1 pasta and 1 drink, out of 3 pasta and 2 drink choices. The set is 50 yen cheaper than ordering the pasta and drink separately.

Given the pasta and juice prices of the day, create a program to find out the minimum price of the lunch set for that day.

**INPUT**

The input has 5 rows, with 1 integer written in each row.

In the first row, the price for the first pasta choice is written.

In the second row, the price for the second pasta choice is written.

In the third row, the price for the third pasta choice is written.

In the fourth row, the price for the first drink choice is written.

In the fifth row, the price for the second drink choice is written.

Given the input data, ensure that the price for each item is in the range of 100-2000 yen.

**OUTPUT**

Output the minimum price of the lunch set for that day in one row.

**INPUT/OUTPUT EXAMPLES**

**INPUT 1**

800

700

900

198

330

**OUTPUT 1**

848

**INPUT 2**

1999

1999

100

189

100

**OUTPUT 2**

150

In INPUT EXAMPLE 1, a set consisting of the second pasta and the first drink will cost 700 + 198 - 50 = 848, which is the minimum price of the day.

In INPUT EXAMPLE 2, a set consisting of the third pasta and the second drink will cost 100 + 100 - 50 = 150, which is the minimum price of the day.

※Save each example's input/output data as a file by right-clicking the links.

**Soccer**

**QUESTION**

Soccer is a popular sport in JOI Kingdom, and a soccer event known as the JOI League takes place every week.

N teams belong to the JOI League, numbered from 1 to N. Each team will play once against every other team in the League. In other words, there will be a total of [N × (N - 1) / 2] matches. The results of each match will determine how many points a team gets. The winner of a match gets 3 points, while the losing team gets 0 points. In a draw, both teams will get 1 point. The final ranking will be determined by the total number of points gotten. The more points a team has, the higher their ranking.

For example, if there are 4 teams in the League, they will play 4 × (4 - 1) / 2 = 6 matches in total. The results are shown in the table below. The number on the left of the hyphen is the number of goals scored by the team in the row, while the number on the right of the hyphen is the number of goals scored by the team in the column.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Team 1 | Team 2 | Team 3 | Team 4 | Number of wins | Number of losses | Number of draws | Total points |
| Team 1 | --- | 0 - 1 | 2 - 1 | 2 - 2 | 1 | 1 | 1 | 4 |
| Team 2 | 1 - 0 | --- | 1 - 1 | 3 - 0 | 2 | 0 | 1 | 7 |
| Team 3 | 1 - 2 | 1 - 1 | --- | 1 - 3 | 0 | 2 | 1 | 1 |
| Team 4 | 2 - 2 | 0 - 3 | 3 - 1 | --- | 1 | 1 | 1 | 4 |

In this situation, Team 2 is in first place with the highest number of points.

Teams 1 and 4 have the same number of points, so they both take second place.

Team 3, with the lowest number of points, is in fourth place.

Given the results of all matches, create a program to find out the final ranking of each team.

**INPUT**

In the first row, the number of teams, N, where 2 ≦ N ≦ 100, is written.

For the next [N × (N - 1) / 2] rows, the results for each match are written. In the (i + 1)th row, where 1 ≦ i ≦ N × (N - 1) / 2, 4 integers are written with a space between them.

* Ai, where 1 ≦ Ai ≦ N. This represents the team that is competing in the ith match.
* Bi, where 1 ≦ Bi ≦ N. This represents the other team that is competing in the ith match.
* Ci, where 0 ≦ Ci ≦ 100. This represents the number of goals scored by team Ai.
* Di, where 0 ≦ Di ≦ 100. This represents the number of goals scored by team Bi.

Ensure that Ai ≠ Bi, and that each pair of teams is not written more than once.

**OUTPUT**

The output has N rows.

In the ith row, where 1 ≦ i ≦ N, output the final ranking of team i.

**INPUT/OUTPUT EXAMPLES**

**INPUT 1**

4

1 2 0 1

1 3 2 1

1 4 2 2

2 3 1 1

2 4 3 0

3 4 1 3

**OUTPUT 1**

2

1

4

2

**INPUT 2**

5

1 2 1 1

3 4 3 1

5 1 1 2

2 3 0 0

4 5 2 3

1 3 0 2

5 2 2 2

4 1 4 5

3 5 4 0

2 4 0 1

**OUTPUT 2**

2

4

1

4

3

INPUT EXAMPLE 1 is the example given in the question section.

In INPUT EXAMPLE 2, the results are shown below.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Number of wins | Number of losses | Number of draws | Total points |
| Team 1 | 2 | 1 | 1 | 7 |
| Team 2 | 0 | 1 | 3 | 3 |
| Team 3 | 3 | 0 | 1 | 10 |
| Team 4 | 1 | 3 | 0 | 3 |
| Team 5 | 1 | 2 | 1 | 4 |

※Save each example's input/output data as a file by right-clicking the links.

**Best Pizza**

**QUESTION**

Manager K is a regular customer at JOI Pizza Shop in JOI City. Due to some circumstances, he decided to start living a more frugal lifestyle. Thus, when he orders a pizza, he hopes to get the maximum number of calories per dollar spent. He calls such a pizza the 'best pizza'. There may be more than 1 'best pizzas'.

At JOI Pizza Shop, customers can choose from N kinds of toppings to add to their pizza base. The same topping cannot be chosen more than once. Pizzas without any toppings can also be ordered. The price of the pizza base is A dollars, while the price of each topping is B dollars. The price of the pizza is the total price of the base and all chosen toppings. In other words, a pizza with k kinds of toppings, where 0 ≦ k ≦ N, will cost (A + k × B) dollars. The pizza's calories are the calories of the base plus all chosen toppings.

Given the prices and calories of the pizza base and all toppings, create a program to find out the number of calories per dollar of a 'best pizza'.

**INPUT**

The input has (N + 3) rows.

In the first row, the number of kinds of toppings available, N, where 1 ≦ N ≦ 100, is written.

In the second row, 2 integers are written with a space between them.

* A, where 1 ≦ A ≦ 1000. This represents the price of the pizza base.
* B, where 1 ≦ B ≦ 1000. This represents the price of a topping.

In the third row, the number of calories of the pizza base, C, where 1 ≦ C ≦ 10000, is written.

In the (3 + i)th row, where 1 ≦ i ≦ N, the number of calories of topping i, Di, where 1 ≦ Di ≦ 10000, is written.

**OUTPUT**

Output the number of calories per dollar of a 'best pizza' in one row. Remove all decimals to ensure that the output is an integer.

**INPUT/OUTPUT EXAMPLES**

**INPUT 1**

3

12 2

200

50

300

100

**OUTPUT 1**

37

**INPUT 2**

4

20 3

900

300

100

400

1300

**OUTPUT 2**

100

In INPUT EXAMPLE 1, adding Toppings 2 and 3 will give a total of 200 + 300 + 100 = 600 calories. Thus, the pizza will cost 12 + 2 × 2 = 16 dollars.

For this pizza, the calories per dollar is 600 / 16 = 37.5 calories. By removing decimals, the output is 37.

**Pasta**

**QUESTION**

You love pasta and cook it for dinner every night. You can make pasta with 3 types of sauces: tomato sauce, cream sauce, and basil sauce.

You are planning dinner for the next N days. You will choose one type of pasta for each day. However, eating the same type of pasta too much makes you bored, so you will not choose to make the same pasta for more than 3 consecutive days. In addition, the pasta you will make for K out of N days has already been decided.

Given the value of N and information regarding K days, create a program that fulfils all the above conditions, dividing the result by 10000 and taking the remainder as the final output.

**INPUT**

The input has (K + 1) rows.

In the first row, two integers are written with a space between them.

* N, where 3 ≦ N ≦ 100.
* K, where 1 ≦ K ≦ N.

In the (1 + i)th row, where 1 ≦ i ≦ K, two integers are written with a space between them.

* Ai, where 1 ≦ Ai ≦ N. This represents that the pasta on Day Ai has already been decided. The integer for Ai cannot be repeated.
* Bi, where 1 ≦ Bi ≦ 3. Bi = 1 represents tomato sauce, Bi = 2 represents cream sauce, while Bi = 3 represents basil sauce.

Ensure that there is at least 1 possible output, given the input data.

**OUTPUT**

Calculate the number of possible situations that fulfils all the conditions, dividing the result by 10000 and taking the remainder as the final output, in one row.

**INPUT/OUTPUT EXAMPLES**

**INPUT 1**

5 3

3 1

1 1

4 2

**OUTPUT 1**

6

**INPUT 2**

20 5

10 2

4 3

12 1

13 2

9 1

**OUTPUT 2**

2640

In INPUT EXAMPLE 1, you are planning for 5 days. You will make tomato sauce pasta on Days 1 and 3, and cream sauce pasta on Day 4. Considering that you will not choose to make the same pasta for more than 3 consecutive days, the number of situations that fulfils this condition is 6.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 |
| Plan 1 | 1 | 2 | 1 | 2 | 1 |
| Plan 2 | 1 | 2 | 1 | 2 | 2 |
| Plan 3 | 1 | 2 | 1 | 2 | 3 |
| Plan 4 | 1 | 3 | 1 | 2 | 1 |
| Plan 5 | 1 | 3 | 1 | 2 | 2 |
| Plan 6 | 1 | 3 | 1 | 2 | 3 |

In this table, 1 represents tomato sauce, 2 represents cream sauce, while 3 represents basil sauce.

In INPUT EXAMPLE 2, there are a total of 4112640 situations that fulfil the conditions. Dividing that by 10000, the output is a remainder of 2640.

※Save each example's input/output data as a file by right-clicking the links.

**Illumination**

**QUESTION**

JOI Company is composed of hexagonal office buildings with 1-meter sides, as shown in the diagram. As Christmas is approaching, the sides of the buildings will be decorated with illuminations. Since decorating sides that cannot be seen from outside is a waste of money, illuminations will only be installed on sides of hexagons which are not facing another building.

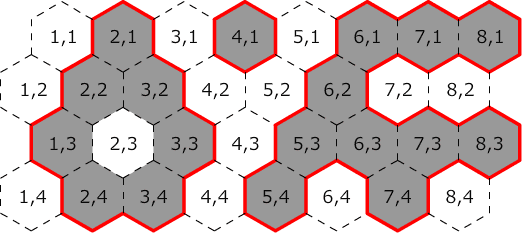


Diagram: An example of the arrangement of JOI Company's office buildings

The diagram above shows an example of JOI Company's office buildings as seen from above. Each hexagon is labeled with numbers. Hexagons that are grey have a building on it, hexagons that are white have no building built on it. The red lines show sides where illuminations will be installed. The total length of those sides is 64 meters.

Given the map of JOI Company's office buildings, create a program to find out the total length of sides where illuminations will be installed. Assume that there are no buildings outside the perimeter of the map.

**INPUT**

In the first row, two integers are written with a space between them.

* W, where 1 ≦ W ≦ 100.
* H, where 1 ≦ H ≦ 100.

For the next H rows, in the (i + 1)th row, where 1 ≦ i ≦ H, a string with W integers is written with a space between them.

The jth integer designates a hexagon at position (j, i). j = 1 when there is a building built on that hexagon, while j = 0 when there is no building built on that hexagon. Ensure that there is at least 1 building in the input data.

The map must follow the rules below:

* The most northwestern hexagon is designated (1, 1).
* Hexagon (x, y) is adjacent to the west of hexagon (x + 1, y).
* When y is an odd number, hexagon (x, y) is adjacent to the northeast of hexagon (x, y + 1).
* When y is an even number, hexagon (x, y) is adjacent to the northwest of hexagon (x, y + 1).

**OUTPUT**

Output the total length of sides where illuminations will be installed in one row.

**INPUT/OUTPUT EXAMPLES**

**INPUT 1**

8 4

0 1 0 1 0 1 1 1

0 1 1 0 0 1 0 0

1 0 1 0 1 1 1 1

0 1 1 0 1 0 1 0

**OUTPUT 1**

64

**INPUT 2**

8 5

0 1 1 1 0 1 1 1

0 1 0 0 1 1 0 0

1 0 0 1 1 1 1 1

0 1 0 1 1 0 1 0

0 1 1 0 1 1 0 0

**OUTPUT 2**

56

In INPUT EXAMPLE 1 is the same as the diagram given in the question section. The total length of sides where illuminations will be installed is 64 meters.

※Save each example's input/output data as a file by right-clicking the links.

**Zig-Zag Numbers**

**QUESTION**

When a string of positive integers (without leading zeros) is written, and each successive digit follows a pattern of being higher, then lower than the previous one, the string of numbers is known as a 'zig-zag number'. For example, the number 2947 has digits 2 → 9 → 4 → 7, which follow the pattern of higher → lower → higher, so it is a zig-zag number. 71946 also has a pattern of lower → higher → lower → higher and is a zig-zag number too. On the other hand, the numbers 123 and 71446 and 71442 and 88 are not zig-zag numbers. Now, you will determine if a string of positive integers is a zig-zag number.

Create a program to find out the number of possible zig-zag numbers in a range of A to B that are multiples of M, dividing the result by 10000 and taking the remainder as the final output.

**INPUT**

The input has 3 rows, with integers A, B, and M written in each, where 1 ≦ A ≦ B ≦ 10500 and 1 ≦ M ≦ 500.

※Note that there is a chance that the values of A and B cannot fit in the data type representing normal integers.

**OUTPUT**

Calculate the number of possible zig-zag numbers in the range of A to B that are multiples of M, dividing the result by 10000 and taking the remainder as the final output in one row.

**INPUT/OUTPUT EXAMPLES**

**INPUT 1**

100

200

5

**OUTPUT 1**

13

**INPUT 2**

6

1234567

3

**OUTPUT 2**

246

In INPUT EXAMPLE 1, there are 13 zig-zag numbers in the range of 100 to 200 that are multiples of 5: 105, 120, 130, 140, 150, 160, 165, 170, 175, 180, 185, 190, and 195.

In INPUT EXAMPLE 2, there are 50246 zig-zag numbers in a range of 6 to 1234567 that are multiples of 3. Dividing that by 10000, the output is a remainder of 246.

※Save each example's input/output data as a file by right-clicking the links.