

Problem Set 2

1 Road Maintenance

Task: You start your new job as manager of the road maintenance depot of a district in Manhattan. Your district has the shape of a polygon, and the road system has the shape of a perfect grid (so the Broadway is located in a different district). To get an overview you want to count all the crossroads that you will be responsible for in the future.

Input: The input starts with a line containing the number m of corners of the polygon. Then, m lines follow; where the i -th line contains the x - and y -coordinate of the i -th corner. The coordinates are all integers.

You may assume that the polygon is non-self-intersecting, but it is possible that more than 2 corners are located on a line.

Output: Output the number of crossroads that are located in your district. You are also responsible for the crossroads that lie on the boundary.

Sample input:

```
5
0 1
2 0
0 -2
-1 -1
-2 0
```

Sample output:

```
10
```

2 k -means II

Task: A crucial part of the implementation of many clustering algorithms is finding the closest center to a point. In this task, we consider a special case of this problem where the points and centers are 1-dimensional, i.e., $P \subseteq \mathbb{R}$ and $C \subseteq \mathbb{R}$. Your task is to solve the following problem: For each query point $x \in \mathbb{R}$, decide whether the closest point in C is at distance at most R from x . If so, report the point closest to x .

Input: Each input file starts with a line with two integers and a floating point number. The first number c (integer) is the number of centers, the second number q (integer) is the number of queries, and the third number is R (floating point number with two positions after the point). In the following c lines, each line has a floating point number (two positions after the point) representing a center. The subsequent q lines contain a floating point number (two positions after the point) each, representing a query point.

Output: For each of the q query points, write a line with **none in range** if the closest point is at distance $> R$, and otherwise, write a line with the closest center (with two positions after the point). If there are two closest centers $c_1, c_2 \in \mathbb{R}$, choose the smaller one.

Sample Input:

```
2 3 1.00
0.00
1.00
1.10
0.50
10.00
```

Sample Output:

```
1.00
0.00
none in range
```

3 k -means III

Task: Now the task is to solve a more realistic version of the last task. The points now live in \mathbb{R}^d with the Euclidean metric, and in addition, the task is dynamic: Instead of getting all centers upfront, we have to produce centers on the fly. More precisely, do the following: Read a stream of points. Compute a set of centers C ; in the beginning, C is empty. For each query point $x \in \mathbb{R}^d$, decide whether the closest point in C is at distance at most R from x . If so, ignore x . If not, add x to C .

Input: The first line contains two numbers. The first number $d \leq 100$ is an integer, the dimension of the points. The second number is a floating point number with three positions after the point, it is the threshold value R . After that, a (not previously specified) number of lines follows, each containing a point, given by d numbers separated by spaces. The numbers are floating points with three positions after the point.

Output: Output a single integer: The number of points that are added to the center set.

Sample Input:

```
2 1.000
0.000 0.000
1.000 1.000
0.500 0.500
2.000 1.000
2.500 0.500
```

Sample Output:

```
3
```

4 Basic Math

Task: You are given $U, V, W \in \{1, \dots, 400000\}$. We define

$$S = \{(x, y, z) \in \mathbb{Z}^3 : (x \neq y) \wedge (x \neq z) \wedge (y \neq z) \\ \wedge (x + y + z = U) \wedge (xyz = V) \wedge (x^2 + y^2 + z^2 = W)\}.$$

Output the lexicographically smallest element in S . In case that S is empty, print “empty set”. A triple $s := (x, y, z)$ is lexicographically smaller than a triple $s' := (x', y', z')$ if the first coordinate where s and s' differ is smaller in s . More precisely, s is lex. smaller than s' if:

- $x < x'$,
- otherwise $x = x'$, but $y < y'$
- otherwise $x = x'$ and $y = y'$, but $z < z'$.

Remark: Even though it is possible to solve this problem in `Python`, it is recommended to use `C++` or `Java`.

Input: There will be several test cases concatenated in one file. The first line contains a positive integer n that specifies the number of test cases. In each of the following n lines a test case will be described. Each test case will be described by three integers $U, V, W \in \{1, \dots, 400000\}$.

Output: For each test case print in a single line x, y and z . If S is empty, print “empty set”.

Sample Input:

```
3
1 1 1
88 1136 5298
42 420 842
```

Sample Output:

```
empty set
1 16 71
1 20 21
```

5 Primary School

You are a teacher at a primary school. This friday, teaching is suspended in favor of a big summer party. Of course the pupils have to attend the party, and all are asked to write their name on a list for contact tracing.

You are pretty sure that the current list is not correct because you have seen children at the party that you do not see on the list, but you have also seen children write down their name multiple times. On top of that, the children do not yet excel at writing and often write the letters of their name in the wrong order.

Now before making an announcement and asking parents to check if their children are on the list, you want to determine how many children actually wrote their name down.

Input: The first line contains the number n of entries on the list. The following line contains n strings with letters in the alphabet $\{a, \dots, z\}$ (all lower case). You may assume that the children always write the correct set of letters for their name (just possibly in the wrong order), that the names of the children are unique, and even more: The multiset of letters in each child's name is unique, i.e., if we pick two names and write them in the wrong order, the two resulting names will always still be different. (For example, Enna and Anne will not both be at this school.)

Output: Output the number of children that have written their name on the list at least once.

Sample Input:

```
4
carsten einalem crasten andreas
```

Sample Output:

```
3
```

6 Rectangular Fields

Farmer William has an old, rectangular orchard of pear trees. Originally, it was completely filled with trees, but over the years many trees had to be felled. Now, William wants to use the free space to keep sheep. He has the brilliant idea to use the pear trees as corners for the pasture's fence. He wants to change the sheep's pasture every day. In addition, every day's pasture should be rectangular. How many days can William go without repeating any one pasture?

Input: The first line of the input has the number r of rows of the orchard. Every following line consists of a string in $\{0, 1\}^r$. The position of pear trees are marked with 1. You can assume that r is bounded from above by 2000.

Output: The number of subgrids (axis-parallel rectangles) such that all corners are marked with a 1.

Remark: The organizer is not aware of a solution in **Python** that adheres to the time limit. It is recommended to solve the problem in **C++** or **Java**.

Sample Input:


```
3
111
110
011
```

Sample Output:

```
2
```

7 Treasure Hunt

In your new favorite mobile game, you are controlling a little robot that collects gem stones on a rectangular field with n rows each containing m squares (the values of n and m depend on the level that you have reached). The robot starts in the upper left corner at position $(1, 1)$ and has to end at position (n, m) . You may only move him down or to the right in each step, i.e., if we denote the robots position in row i , column j by (i, j) , you may move him to $(i + 1, j)$ or to $(i, j + 1)$, respectively. At each position that the robot reaches, he collects all gem stones that lie at that position. Your goal is to collect as many gem stones as possible.

	100	2	3	5	3	5	3
2	100	100	100	3	2	5	4
3	5	0	100	2	2	2	4
3	4	3	100	1	2	1	5
1	0	1	100	100	100	2	5
1	4	1	4	5	100	100	100
1	4	0	4	4	1	1	2

Input: The first line contains the integer $n \in \{1, \dots, 1000\}$, the second line contains the integer $m \in \{1, \dots, 1000\}$. The following n lines contain m non-negative integers each. In the i th of these lines, the j th number is the number of gems at position (i, j) .

Output: An integer, the maximum number of gem stones that may be collected.

Sample Input:

```

3
4
1 12 22 5
0 0 6 0
1 8 0 1

```

Sample Output:

```

42

```

8 Fridays for Future

This week, you want to organize a *Fridays for Future* protest in your home town. In order to reach the largest amount of people, you want to position groups of protesters with a huge poster at intersections in such a way that everyone traveling within the city passes at least one poster. Here, traveling means starting at an intersection and driving to a different intersection.

You know that you have a large number of fellow protesters, but it is quite expensive to print large posters. Ideally, you would like to minimize the number of posters required. Unfortunately, there is not much time left until the next friday and a friend that studies computers science told you that your problem will likely take a very long time to solve exactly. So you accept the next best thing and are happy if you print at most double the minimum number of posters required.

Input: Your home town is given as a graph where roads are represented by edges and intersections are represented by nodes. The first line of the input contains two numbers, the number of nodes n and the number of edges m . This is followed by m lines containing two integers $a, b \in \{0, \dots, n-1\}$. Each such pair describes an edge in the graph.

Output: The number of posters required so that it is impossible to travel along any path (of at least one edge) in the graph without seeing a poster. This number does not have to be the minimal number required, but it may not deviate by factor of more than 2 from this number.

Sample Input:

```
5 5
0 2
1 2
1 3
2 3
3 4
```

Sample Output:

```
2
```

While 2 is the optimum solution, values up to 4 would be fine as well.