C++ Advanced – Regular Exam – 16 June 2024

Please submit your source code to all below-described problem in Judge.

3. Traffic

You're designing a smart traffic control system. To do so, you need to write a helper program first.

You have to handle between two to twenty-six interceptions, named from **A** till **Z**. Your **first parameter** in the input is an **integer number**, which give you **the number of interceptions**. For example, **3** means that the interceptions are from **A** till **C**.

Then you have the next part of the input data: the distance between each one of the iterceptions. The distance is always a single digit from 1 to 9. All distances are given via matrix. For the example above, the matrix looks like:

```
0 3 4
3 0 1
6 2 0
```

This describes the following distances:

	A	B	C
A	0	3	4
B	3	0	1
C	6	2	0

Please note: it is normal the distance between **A** and **C** to be **4**, but between **C** and **A** to be **6**. In the above matrix, the differences are as follows:

```
AB is 3 (row A, column B)
AC is 4 (row A, column C)
BA is 3 (row B, column A)
BC is 1
CA is 6
CB is 2
```

Your first task is to read the matrix, then print out the matrix with rows decsriptions and column titles, as shown above (and in the examples below).

Then you must **analyze the matrix**, and determine all **"optimization opportunities"**. An optimization opportunity is a case, where there is difference between the roads of two points. For example, in the above matrix, **AC** is **4**, but **CA** is **6**, so the **optimization opportunity** is the absolute number of their difference: **2**. You must find and print out all optimization opportunities, using the following format. For the example above, it is as follows:

```
AC(4) - CA(6)
BC(1) - CB(2)
```

Then you must print the sum of all optimization opportunities, e.g.:

```
Total opportunities: 3
```

You must also determine the maximum (biggest) optimization opportunity, in the example above it's "AC(4) - CA(6)" with value of 2, and print it as follows:

```
Max optimization: 2: AC-CA
```

If there are more than one optimization opportunities matching the max optimization, you must prin all of them, comma-separated. For example (this is **not** from the example above):

```
Max optimization: 2: AC-CA, DF-FD
```

Hints:

1. The diagonals of the distances matrix will always be `0`, because the distance between a crossroad and itself is always `0`.
2. The data will **always be correct**, and there will be always **at least two interceptions** in the matrix.
3. There can be no optimization opportunities.
   In this case you output no opportunities, and the output for the biggest optimization must be
   `Max optimization: none`

Example 1

Input	Output	Explanation
3 0 3 4 3 0 1 6 2 0	A B C A 0 3 4 B 3 0 1 C 6 2 0	It will be nice, if the spaces match. The first printed row is the matrix' column titles, then you follow up with each row, starting with the row letter, space, then each one of the distances for the given row, etc.
	AC(4) - CA(6) BC(1) - CB(2)	There are two optimization opportunities for the matrix above. The distance between AC is 4, but between CA is 6, which gives is abs(4-6) = 2, and the distance between BC is 1, and between CB is 2, which gives us abs(1-2) = 1.
	Total opportunities: 3	The sum of all optimization opportunities is 2+1 = 3.
	Max optimization: 2: AC-CA	The maximum optimization is 2. There's only one path with the maximum optimization value: the path between AC and CA.

Example 2

Input	Output
4 0 3 5 1 1 0 4 6 7 3 0 1 4 3 4 0	A B C D A 0 3 5 1 B 1 0 4 6 C 7 3 0 1 D 4 3 4 0 AB(3) - BA(1) AC(5) - CA(7) AD(1) - DA(4) BC(4) - CB(3) BD(6) - DB(3) CD(1) - DC(4)

	Total opportunities: 14 Max optimization: 3: AD-DA BD-DB CD-DC

Example 3

Input	Output	Explanation
5 0 3 4 7 9 3 0 1 3 1 4 1 0 4 2 7 3 4 0 5 9 1 2 5 0	A B C D E A 0 3 4 7 9 B 3 0 1 3 1 C 4 1 0 4 2 D 7 3 4 0 5 E 9 1 2 5 0 Total opportunities: 0 Max optimization: none	After printing out the matrix, there are no optimization opportunities in the distances, as the distance between each point in both directions are the same.