Problem 24: OSPF Routing

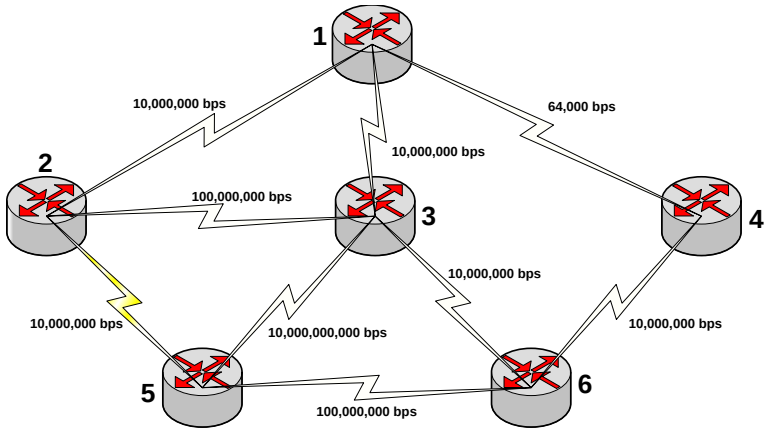
Source filename:	`ospf.(cpp|java)`
Input filename:	`ospf.in`
Output filename:	`ospf.out`

OSPF (Open Shortest Path First) is a well-known routing protocol that uses Dijkstra's shortest path algorithm to make routing decisions within an OSPF *area*. An OSPF *area* is a group of contiguous networks and routers in each router shares link-state information with all the other routers in the area. Link-state information from a router includes the status of all interfaces connected to the given router and the *cost* of transmitting packets from every router in the OSPF area to every other router in the OSPF area. One well-known manufacturer of routers defines the *cost* of transmitting a packet via a given interface by the formula: `cost` = $\frac{10^8}{bandwidth}$, where *bandwidth* is measured in bits per second (bps). For example a *bandwidth* of 10,000,000 bps would yield a `cost` value equal to 10. The value of `cost` should be rounded to two decimal places (nearest hundredth).

The input file for this problem contains 1 or more data sets. The first line in a data set contains a single integer, n ($0 \leq n \leq 100$), where n represents the number of routers in an OSPF area. A value of 0 for n indicates the end of the input file.

The next n lines will contain n non-negative integers separated by one or more spaces. This n x n array of integers represents the adjacency matrix for the graph that represents the OSPF area. Each row and each column represent one of the routers in the OSPF area. The non-negative integer stored in row i and column j represents the bandwidth (bps) of the link between router i and router j. The value of a given bandwidth will not exceed 10,000,000,000 bps. A value of 0 indicates the absence of any direct connection between the two routers. The values along the main diagonal of the matrix (where row = column) will all be zero as well. Notice that the array will be symmetric with respect to the main diagonal.

For instance, the input values for the following OSPF area are shown below.




```
6
0           10000000    10000000    64000       0           0
10000000    0           100000000   0           10000000    0
10000000    100000000   0           0           10000000000 10000000
64000       0           0           0           0           10000000
0           10000000    10000000000 0           0           100000000
0           0           10000000    10000000    100000000   0
```

After the n^{th} line of the array, there will be one or more lines each of which contains a single integer, k ($0 \leq k \leq n$). A zero indicates the end of the given data set. For each k value, the output file should contain a line that contains the number k, followed by a colon, a space, and a list of all of the other router numbers (in numerical order) along with the total cost associated with the shortest path from router k to each of the routers, enclosed in parentheses. A single space should separate each router (and its corresponding total cost) from the next router number. For instance, for the data set above, if the next line contained the integer 2, the corresponding output line should be:

```
2: 1(10.00) 3(1.00) 4(12.01) 5(1.01) 6(2.01)
```

Between the lines of output associated with different data sets, output one line containing 5 asterisks.

Example Input File (`ospf.in`)

```
6
0           10000000    10000000    64000       0           0
10000000    0           100000000   0           10000000    0
10000000    100000000   0           0           10000000000 10000000
64000       0           0           0           0           10000000
0           10000000    10000000000 0           0           100000000
0           0           10000000    10000000    100000000   0
2
3
0
5
0 14285714 25000000 16666667 100000000
14285714 0 50000000 33333333 0
25000000 50000000 0 20000000 0
16666667 33333333 20000000 0 100000000
100000000 0 0 100000000 0
1
0
0
```

Example Output File (`ospf.out`)

```
2: 1(10.00) 3(1.00) 4(12.01) 5(1.01) 6(2.01)
3: 1(10.00) 2(1.00) 4(11.01) 5(0.01) 6(1.01)
******
1: 2(5.00) 3(4.00) 4(2.00) 5(1.00)
```