2002 Sun Microsystems and TopCoder Collegiate Challenge – Problem Statement

```
Roadmap PROBLEM STATEMENT
You will be given a String[] network, which describes the road network between
intersections, as follows:
*each element of network will be formatted (quotes added for clarity):
"#X:#1,#2,...,#n"
Where #X is the origin intersection number, and the comma-delimited #s that
follow the colon are all intersections such that there is a direct (no
intermediate intersection) ONE-WAY street between intersection #X and
intersection #i, where i is the ith intersection in the list (i=[1..n],
inclusive).

Given this network and an integer representing the originating intersection,
return how many intersections are accessible from that intersection (including
itself).

DEFINITION
Class name: Roadmap
Method name: numRoutes
Parameters: String[], int
Returns: int
The method signature is:
int numRoutes(String[] network, int intersection)
Be sure your method is public.

TopCoder will ensure the following:
*network will contain between 1 and 50 elements, inclusive.
*each element of network will contain between 3 and 50 characters, inclusive.
*each element of network will be formatted as above, with the following
constraints:
  *each intersection number is between 1-999, inclusive.  There will be no
leading zeros.
  *no intersection will have a direct route to itself.
  *each intersection will have at least 1 direct route (i.e. there will be no
elements formatted "#:").  If an intersection has no outgoing roads, it will
simply not appear as an element in network.
  *each comma-delimited intersection will be unique within the String.
  *each origin will be unique.  That is, no two origins in the problem will be
the same.
  *there will be no spaces
*intersection will be an integer between 1 and 999, inclusive, and will appear as
either an origin or a destination in network.

NOTES:
-An intersection is accessible from itself, and therefore counts in the total.

EXAMPLES
1)
network={
"1:2,3,4,5,6,7,8,9",
"2:10,11"
}
intersection = 1
return = 11

2)
```

```
network={
"1:2,3,4,5,6,7,8,9",
"2:10,11"
}
intersection = 4
return = 1

3)
network={
"1:2,3",
"4:5,6",
"7:8",
"3:4"
}
intersection = 1
return = 6

4)
network={
"1:2",
"2:1"
}
intersection = 1
return = 2

5)
network={
"1:2,4,6,7,8,10",
"2:5,6,8,9,11,15,16",
"3:4,6,7,8",
"4:5",
"6:8,9,13",
"8:9,10,13,41",
"13:14",
"41:3"
}
intersection = 1
return = 16

6)
network={
"1:2,4,6,7,8,10",
"2:5,6,8,9,11,15,16",
"3:4,6,7,8",
"4:5",
"6:8,9,13",
"8:9,10,13,41",
"13:14",
"41:3"
}
intersection = 41
return = 11
```