Problem 10 | **6 Points**

Which way did it go?

Program Name: router.cpp **Input File: router.dat**

One of the most important functions in the I/T world today is the efficient routing of packets in networks. Typically, this routing occurs using a simple internet protocol (IP) address. IP addresses are 4 bytes long and are expressed as 4 numbers ranging 0-255 (a single unsigned byte is 0-255) separated by periods (example 9.87.65.210). When a router receives a packet, it uses the destination IP address from the packet to examine a routing table and determines the next "hop" for the packet.

A routing table contains ranges of IP addresses that are to be sent to each peer router or machine on the same subnet. Network topologies are typically constructed such that the whole ranges of IP addresses within the domain of a single byte can be routed to a single peer. This is possible by setting up "routing masks" that use wild cards to match any number in one or more of the IP bytes. For example, it is possible that all addresses starting with 125.14 are routed to a single peer. The entry in the routing table would contain wild cards in the routing range in an address that appears as "125.14.*.*". Your router will include this functionality and will search the routing table and select the first match it finds.

Rule #	Routing Mask	Routing Address	Comments
1	`226.76.*.*`	`226.76.27.1`	Anything starting with 226.76 will go here
2	`143.*.*.*`	`98.24.52.1`	Anything starting with 143 will go here
3	`115.74.*.*`	`115.74.*.1`	Anything starting with 115.74 will go to a router ending in 1 with the same third number (ie 115.74.99.16 will go to 115.74.99.1)
4	`225.8.15.*`	`225.8.*.1`	Although it is odd to have an absolute value (15) in the 3rd value of the mask and an wild card in the same position of the routing address, it is valid because it will always evaluate out to 15.
5	`225.*.*.*`	`225.14.9.1`	Anything that starts with 225 goes to 225.14.9.1
6	`9.*.*.*`	`9.1.1.1`	Anything that starts with a 9 goes to 9.1.1.1
7	`157.19.230.*`	`226.76.27.1`	Anything starting with 157.19.230 goes to 226.76.27.1
8	`27.18.43.18`	`27.18.43.18`	This guy must have done something bad if we are sending all of his stuff back to him.
9	`27.*.*.*`	`27.*.*.1`	Anything starting with a 27 will go to the same address with a 1 in the last position.
10	`187.32.42.*`	`226.75.18.1`	Anything starting with 187.32.42 goes to 226.75.18.1
11	`*.*.*.*`	`226.75.11.1`	Everything else goes to 226.75.11.1 since we don't know what to do with it. Most routing tables have this but it is not required.

You can see from the table above:

1. It is possible for an address to match more than one mask. In this case, your program should select the first table entry (top-to-bottom) that has a matching mask.
2. Wild cards can be used in any position in the mask but it does not make sense to use a wild card in one position and then use absolute values in the positions to the right. (For example, a mask with 225.*.19.17 does not make sense from a network topology point of view.) Therefore, you may assume that such a mask would be invalid and your program will not be subjected to invalid input.
3. It is possible for 1 or more routing addresses to contain a wild card. When this is the case, your program should use the value from the corresponding destination IP address position in constructing the routing address. For example, the address 27.189.212.47 would be routed to 27.189.212.1 per rule #9.

Input

Input to your program will consist of a routing table followed by a series of destination IP addresses. The first line of input will contain a single integer ($1 \leq R \leq 100$) indicating the number of rules in the routing table. The next R lines of input will each contain a routing rule. Routing rules will be of the form "*x.x.x.x y.y.y.y*" where *x.x.x.x* is the routing mask and *y.y.y.y* is the routing address and there is a single space separating them. In all cases, valid values of *x* and *y* are (0≤*x,y*≤255) and "*". All entries of the routing table will start in column 1 and there will be no other input on any of the lines describing the routing table.

The remaining lines of the input file will each contain a single destination IP address of the form "*z.z.z.z*" where the valid values are (0≤*z*≤255). Each destination IP address will start in column 1 and there will be no other input on any destination IP address line. You can also be assured that no value in the input file will contain leading zeroes unless zero is the only digit in the value.

Output

For each destination address in the input file, your program is to determine the routing address for the packet from the routing table and print the routing address on a line by itself. If no routing mask in the entire routing table matches the destination address, your program should print the message "Route to the bit bucket" on a line by itself. Your program should contain no other output, blank lines or extraneous characters.

Example: Input File

```
10
226.76.*.* 226.76.27.1
143.*.*.* 98.24.52.1
115.74.*.* 115.74.*.1
225.8.15.* 225.8.*.1
225.*.*.* 225.14.9.1
9.*.*.* 9.1.1.1
157.19.230.* 226.76.27.1
27.18.43.18 27.18.43.18
27.*.*.* 27.*.*.1
187.32.42.* 226.75.18.1
9.87.65.210
115.75.87.83
115.74.9.28
143.87.82.81
226.75.76.76
```

Output to screen

```
9.1.1.1
Route to the bit bucket
115.74.9.1
98.24.52.1
Route to the bit bucket
```