ACM Pacific NW Region Programming Contest
11 November 2000

PROBLEM H
SwampNet Routing

Your competitors in the low end routing business are adding firewall type features and your sales department is frantic. They want to add access lists ASAP.

But the marketeers also require that 'wire speed' routing not be compromised and the routing engine is underpowered. Your job is to write a simulator to help answer the performance questions.

Engineering has used a sniffer to record network traffic and extracted the relevant information from the packet headers.

Here are some samples of the simulated traffic packets:

```
 216.35.137.204      0   131.215.211.5      0   1
131.215.139.100 49152  131.215.90.109     53  17
 24.218.179.217 27015    131.215.89.4   2326  17
148.246.129.175  6699  131.215.142.85   1138   6
```

The fields are:

source IP address	source port	destination IP address	destination port	protocol
dotted decimal address	decimal number	dotted decimal address	decimal number	decimal number

The fields are separated by one or more blanks and the first field may be preceded by one or more blanks.

A dotted decimal address is a 32 bit value, high order byte first, with the decimal representation of each 8−bit byte (0..255) separated by a '.' (decimal point). For example, 1.2.255.15 is 0x0102FF0F.

Ports are in the range: 0..65535

Note that ports are only significant for the TCP (6) and UDP (17) protocols. The port fields will be 0 for other protocols.

The protocol is in the range: 0..255

You will be given sample access lists. They will be syntax checked with fields that are separated by one or more blanks. Some example entries:

```
*  1.2.3.4 255.255.0.15 3   5.6.7.8 0.0.255.255 9  Permit
6  0.0.0.0 0.0.0.0 *  131.215.254.254 255.255.255.255 21   deny
```

The fields are:

protocol	source IP address	source mask	source port	destination IP address	destination mask	destination port	action
decimal number or '*'	dotted decimal address	dotted decimal mask	decimal number or '*' or a range	dotted decimal address	dotted decimal mask	decimal number or '*' or a range	'permit' or 'deny' (case not significant)

A range is two decimal numbers separated by '−' with no whitespace between the numbers and the '−'. For example, 5−52 will match any port number between 5 and 52.

In the access list, the protocol and the ports can be specified as '*', which means match any protocol or port.

A dotted decimal mask is the same format as an address. It indicates which bits are significant when comparing addresses. For example, the addresses 123.234.248.14 and 123.234.254.126 match when compared with the mask 255.255.240.15 (0xFFFFF00F). The 1 bits in the mask show which bits in the addresses to compare. Note that a mask of 0.0.0.0 will cause any address pair to match.

The way an access list works is each simulated traffic packet is compared with the access list entries, in the same order those entries were provided, until the packet matches an entry. For a match to occur, the protocol, source and destination addresses (using the respective masks), and ports must all match. In a real router, the action directs the router to accept or reject the packet, but the simulator just keeps track of how many access list entries are compared, and the number of times each action is taken.

For example, if a packet matches the 3rd, 5th, and 8th entries in a 30 entry access list, (with the action of the 3rd entry being 'permit'), then processing that packet will contribute 3 comparisons and add 1 to the number of 'permit' actions taken. In this case the 5th and 8th entries are not examined further. The first entry that matches is the last examined for a given packet.

INPUT:
The input file will be a series of test cases. Each test case consists of two groups of data, separated by a blank line. A blank line also separates test cases. The first group of data in a test case is the access list entries, one per line. There will be no more than 100 access list entries. The next group of data in the test case is the simulated packet traffic data. Each of these lines will be at most 100 characters. The test case is terminated by a blank line or end−of−file.

The input file is **h.dat**.

OUTPUT:
For each test case print 3 numbers: the total count of access list entries compared; the number of simulated packets resulting in the 'permit' action; and the number of simulated packets resulting in the 'deny' action. Put no leading spaces before the first number and exactly 1 space before the 2nd and 3rd numbers.

See the sample input and output on the next page.

Sample input:

```
* 204.69.0.0  255.255.248.0  * 0.0.0.0 0.0.0.0 * permit
*   0.0.0.0 0.0.0.0 *   0.0.0.0 0.0.0.0 *  DENY

204.69.6.230 3072 24.115.85.245 23 6
192.168.215.17 53  204.248.52.7 1243 17
204.69.12.21 2118  152.163.241.11 21 6

* 0.0.0.0 0.0.0.0 * 0.0.0.0 0.0.0.0 * deny

  192.102.199.19 27961  63.20.61.96 65535 17

6 198.138.176.100 255.255.255.255 *  204.69.4.100 255.255.255.255 111
permit
6 0.0.0.0 0.0.0.0 * 204.69.0.0 255.255.240.0 111 deny
6  0.0.0.0 0.0.0.0 * 204.69.0.0 255.255.248.0 80 permit
* 0.0.0.0 0.0.0.0 * 204.69.0.255 255.255.248.255 * deny
* 204.69.0.0 255.255.248.0 * 0.0.0.0 0.0.0.0 * deny
* 127.0.0.0 255.0.0.0 *  0.0.0.0 0.0.0.0 * deny
* 0.0.0.0 0.0.0.0 *  204.69.7.240 255.255.255.240 9990-9999 deny
* 0.0.0.0 0.0.0.0 *  0.0.0.0 0.0.0.0 * permit

204.69.4.87 80 204.69.9.12 6776 6
134.79.112.65 2186  204.69.4.218 21 6
216.112.217.140 2212 204.69.5.255 22 6
198.138.176.100 1053 204.69.4.100 111 6
198.138.176.100 1054 204.69.5.100 111  6
```

Sample output:

```
5 1 2
1 0 1
20 2 3
```