PROGRAMMER'S CHALLENGE
by Bob Boonstra, Westford, MA

ROUTER RULES

This month's Challenge is based on a suggestion by Peter Lewis and is motivated by a real-world problem. A certain university has a B-class IP subnet, let's call it 199.232.*.* (with apologies to the real-world owner of that subnet). The subnet is broken down into 256 networks for the various faculties and departments, each one having 256 IP numbers. So, for example, the computer club might have 199.232.101.*. Our hypothetical university is charged for communications based on volume, so some of these networks are allowed to talk to the outside world, and others are not. Outside access is controlled by programming a router with a sequence of rules, each of which allows or denies access to some subset of IP numbers. A rule consists of a (mask, value, allow) triplet. For example, say the networks (in hex) 01, 03, 41, 43 are allowed out, and all the rest are barred. The rules could be simply:

```
FF, 01, allow
FF, 03, allow
FF, 41, allow
FF, 43, allow
00, 00, deny
```

But this could be simplified to:

```
BD, 01, allow
00, 00, deny
```

Your objective for this Challenge is to quickly generate a small sequence of rules that allows outside network access to only a specified set of networks. The prototype for the code you should write is:

```
enum {kDeny=0, kAllow=1};

typedef struct Rule {
  long mask;
  long value;
  long allow; /* 0 == deny, 1== allow */
} Rule;

long RouterRules(
  long allowedValues[],
  long numAllowedValues,
  long numBits,
  Rule rulesArray[],
  long maxRules
);
```

The array `allowedValues` is the set of `numAllowedValues` networks that are to be given outside network access. All other networks should be denied access. Instead of being limited to 8 bits as in the example above, network values have `numBits` bits. Your code should generate a sequence of rules that provides access to these networks, and no others. The rule sequence should be as short as possible and stored in `rulesArray`, which is allocated by the caller and is of size `maxRules`. Your code should return the number of rules generated, or return -1 if it cannot find a solution no longer than `maxRules`.

Rules will be triggered by the router in the order provided by your solution, and the first rule to fire for a given network will apply. At least one rule must fire for any possible network value. For example, if `numBits==3`, and we want to allow access to networks 0, 2, 3, 6, and 7, you could use the following rules:

```
    3, 1, deny
    6, 4, deny
    7, 7, allow
```

To encourage code that generates both fast and short solutions, the ranking will be based on minimizing the following function of execution time on my 8500/150 and the number of rules generated:
score = (number of rules generated) + (execution time in seconds) / 2

This will be a native PowerPC Challenge, using the latest CodeWarrior environment. Solutions may be coded in C, C++, or Pascal.

Two Months Ago Winner

Congratulations to **WINNER** (Location), for submitting the fastest entry to the ByteCode Interpreter Programmer's Challenge.

Name	size	data	time
Entry	xxx	xxx	xxx
Entry	xxx	xxx	xxx
Entry	xxx	xxx	xxx
Entry	xxx	xxx	xxx
Entry	xxx	xxx	xxx
Entry	xxx	xxx	xxx

TOP 20 CONTESTANTS

Here are the Top 20 Contestants for the Programmer’s Challenge. The numbers below include points awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank	**Name**	**Points**	**Rank**	**Name**	**Points**
1.	XXX	187	11.	XXX	22
2.	XXX	92	12.	XXX	21
3.	XXX	87	13.	XXX	21
4.	XXX	74	14.	XXX	20
5.	XXX	40	15.	XXX	20
6.	XXX	30	16.	XXX	19
7.	XXX	29	17.	XXX	19
8.	XXX	24	18.	XXX	17
9.	XXX	22	19.	XXX	17
10.	XXX	22	20.	XXX	14

There are three ways to earn points: (1) scoring in the top 5 of any Challenge, (2) being the first person to find a bug in a published winning solution or, (3) being the first person to suggest a Challenge that I use. The points you can win are:

1st place	20 points	5th place	2 points
2nd place	10 points	finding bug	2 points
3rd place	7 points	suggesting Challenge	2 points
4th place	4 points		

Here is WINNER's winning solution:

BYTECODE.C

Copyright © 1996 WINNER

```
/*
BLOCK THIS SPACE
```

*/