There are two kinds of people in the world.
People who think there are two kinds of people
and the rest ...
Scenario

- You have got some important work to do
- Like now!
- Someone just sent you one of those brain teaser puzzles.
- You can't ignore it, because maybe, they'll think you can't do it ...
Problem: from Scientific American

- **Facts:**
  - There are 5 houses (along the street) in 5 different colors: blue, green, red, white and yellow.
  - In each house lives a person of a different nationality: Brit, Dane, German, Norwegian and Swede.
  - These 5 owners:
    - drink a certain beverage: beer, coffee, milk, tea and water,
    - smoke a certain brand of cigar: Blue Master, Dunhill, Pall Mall, Prince and blend
    - keep a certain pet: cat, bird, dog, fish and horse.
Problem: continued ...

1. The Brit lives in a red house.
2. The Swede keeps dogs as pets.
3. The Dane drinks tea.
4. The green house is on the left of the white house (next to it).
5. The green house owner drinks coffee.
6. The person who smokes Pall Mall rears birds.
7. The owner of the yellow house smokes Dunhill.
8. The man living in the house right in the center drinks milk.
10. The man who smokes blend lives next to the one who keeps cats.
11. The man who keeps horses lives next to the man who smokes Dunhill.
12. The owner who smokes Blue Master drinks beer.
13. The German smokes Prince.
14. The Norwegian lives next to the blue house.
15. The man who smokes blend has a neighbor who drinks water.
Problem: made worse ...

- Question: Who keeps fish?

BUT ...
The author of this problem said that 98% of the people in the world couldn't solve it.

- Now I was stuck...
Problem space

- Possibilities = $5!^5 = 2.4 \times 10^{10}$
- In terms of bits of info = 34.5

- Each hint reduces possibilities by $4/5$, and thus gives 2.32 bits of info.
- Hints give 34.8 bits of data

- Estimated uncertainty is thus ~ 0 bits
IDEA: Let's iterate through the possibilities.

At each step we check if layout is inconsistent or violates data.

Data: encode attributes as hash, people as array.

Not yet defined attributes are `undef`
```perl
#!perl -w
@housecolour =qw(blue green red white yellow);
@nationality =qw(Brit Dane German Norwegian Swede);
@beverage =qw(beer coffee milk tea water);
@smoke =qw(BlueM Dunhill PaulMaul Prince Blend);
@pet =qw(cat bird fish horse dog);

my @pers= ({hp=>1}, {hp=>2}, {hp=>3}, {hp=>4}, {hp=>5});
```
sub verify {  my @pers= @_;  for my $cat qw(hp hc nat bev smo pet) {    my %verif;    for my $pers (@pers) {      next unless $pers->{$cat};      return 0 if $verif{$pers->{$cat}};      $verif{$pers->{$cat}}=1;    }  }  #  1. The Brit lives in a red house.  {    my $p = getpers(@pers, "nat", "Brit");    if ($p && $p->{hc} ) {      return 0 unless $p->{hc} eq "red";    }  }  #  4. The green house is on the left of the white house  {    my $p1 = getpers(@pers, "hc", "green");    my $p2 = getpers(@pers, "hc", "white");    if ($p1 && $p2 && $p1->{hp} && $p2->{hp} ) {      return 0 unless ($p2->{hp} - $p1->{hp} == -1);    }  }
for (@nationality) {
    $pers[0]->{nat} = $_;
    unless (verify(@pers)) { $pers[0]->{nat} = undef; next }
} for (@housecolour) {
    $pers[0]->{hc} = $_;
    unless (verify(@pers)) { $pers[0]->{hc} = undef; next }
} for (@beverage) {
    $pers[0]->{bev} = $_;
    unless (verify(@pers)) { $pers[0]->{bev} = undef; next }
} for (@smoke) {
    $pers[0]->{smo} = $_;
    unless (verify(@pers)) { $pers[0]->{smo} = undef; next }
} for (@pet) {
    $pers[0]->{pet} = $_;
    unless (verify(@pers)) { $pers[0]->{pet} = undef; next }
} print $p2->{"nat"}," has fish\n";
    $pers[0]->{pet} = undef; }
    $pers[0]->{smo} = undef; }
    $pers[0]->{bev} = undef; }
    $pers[0]->{hc} = undef; }
    $pers[0]->{nat} = undef; }

Running the code

- On: WinXP PII-350 under cygwin perl

```
andy@a1 ~
$ time perl question.pl
German has fish
real    0m2.600s
user    0m2.163s
sys     0m0.100s
andy@a1 ~
```

Iterate through all possibilities, but bail out as early as possible if it doesn't work.

Test values must be `undefined` before failing back to previous case.

Debugging is quite hard: add iterations gradually.