Code Jam to I/O for Women 2015 - Code Jam to I/O for Women

Analysis: Power Levels

This problem was a somewhat contrived mathematical interpretation of a popular Internet meme. We know that it was a little awkward to have outputs with hundreds of exclamation points per line, but we couldn't resist. (The definition of multifactorials in the problem is real, but multifactorials donâ€™t come up too often in practice!)

Like I/O Error, this is mostly an implementation problem. There are only 9000 different multifactorials of 9000. You can calculate them all, find how many digits are in each one, and choose the one with the largest number of digits that is still smaller than the number of digits in the opponentâ€™s power level. But for the Large data set, most of those multifactorials are far too large to fit in the standard data types found in languages like C++ and Java. There were several workarounds for this:

1. Use a language extension like Javaâ€™s BigInteger that handles large numbers.

2. Use a language like Python that naturally handles arbitrarily large integers.

3. Take advantage of a useful programming contest trick: to multiply large numbers without overflowing the result, you can instead add the logarithms of those numbers and then raise the logarithm base to that power. In our case, since we only want the number of digits, you can use log base 10 and then take the ceiling of the sum. This method brings about the usual danger of small errors introduced by floating-point addition, but no multifactorial of 9000 is close enough to a power of 10 for this to pose a problem.

Hereâ€™s a sample Python solution that demonstrates the log addition method.

```
import math

def multifactorial9000(num_exc):
  ans = math.log10(9000)
  curr = 9000
  curr -= num_exc
  while curr > 0:
    ans += math.log10(curr)
    curr -= num_exc
  return math.ceil(ans)

# from number of exclamation points to number of digits
digits = [-1]*9001
for num_exc in range(1, 9001):
  digits[num_exc] = multifactorial9000(num_exc)

def solve(numdigits):
  i = 1
  while i <= 9000 and numdigits <= digits[i]:
    i += 1
  if i > 9000:
    return "..."
  return "IT'S OVER 9000" + "!"*i

t = int(raw_input().strip())
for case in range(1, t+1):
```

```
  d = int(raw_input().strip())
  print "Case #" + str(case) + ": " + str(solve(d))
```

By the way, the title is a subtle pun: the problem revolves around numbers of digits, which are closely related to powers of 10.