Some answers to the homework questions

Given to students of the New Jersey Governor's School of Engineering and Technology, July 2002
School of Engineering, Rutgers University, Piscataway, New Jersey

Question #1 What are the first 5 decimal digits of 2^{300}? What are the last 5 decimal digits of 2^{300}?

Answer I used the following `Maple` "dialog":

```
>2^300;
2037035976334486086268445688409378161051468393665936250636140449354381299763336706183397376
```

The first 5 digits are 20370 and the last 5 digits are 97376.

Question #2 How many decimal digits does $2^{(2^{(2^{(2^2)})})}$ have? What about $(((2^2)^2)^2)^2$? (I think the first number is the largest you can get with five 2's and exponentiation and parenthesizing and the second is the smallest.)

Answer You *could* ask `Maple` to compute numbers and then count digits. This would be tiresome, and there would likely be errors. Logarithms can be used. You would need to investigate using the `help` command. The default logarithm is the "natural" one, base e. To see how many digits (base 10) a number N has, you would need to compute $\frac{\log(N)}{\log(10)}$ and round up to the nearest integer. (If you don't see why, please learn more about logarithms.) For example, the computation

```
>evalf(log(34)/log(10));
1.531478917
```

shows that 34 has 2 decimal digits. `evalf` is needed to force `Maple` to compute a decimal approximation to the numbers, otherwise `Maple` will just return the answer $\frac{\mathtt{ln(34)}}{\mathtt{ln(10)}}$ since `Maple` is a program and will do what you ask.

So now:

```
>evalf(log(2^(2^(2^(2^2))))/log(10));
19728.30180
>evalf(log((((2^2)^2)^2)^2)/log(10));
4.816479931
```

and the first number has 19,729 decimal digits, while the second has 5 (its value is 65536).

Question #3 Here is a secret-sharing example **mod 17**. Find the constant term mod 17 of a fourth degree polynomial P so that

P(1)=2 P(2)=4 P(3)=6 P(4)=8 P(10)=15

The number should identify one of the animals in the table above.
Comment Do `all` computations mod 17. `Maple` is "smart" enough. This will reduce the work needed.

Answer First I asked `Maple` to compute (and remember, by appropriate labeling!) some Lagrange interpolation polynomials (**mod 17**).

```
>A:=((x-2)*(x-3)*(x-4)*(x-10))/((1-2)*(1-3)*(1-4)*(1-10)) mod 17;
A := 6 (x + 15) (x + 14) (x + 13) (x + 7)
>B:=((x-1)*(x-3)*(x-4)*(x-10))/((2-1)*(2-3)*(2-4)*(2-10)) mod 17;
B := (x + 16) (x + 14) (x + 13) (x + 7)
>C:=((x-1)*(x-2)*(x-4)*(x-10))/((3-1)*(3-2)*(3-4)*(3-10)) mod 17;
C := 11 (x + 16) (x + 15) (x + 13) (x + 7)
>DD:=((x-1)*(x-2)*(x-3)*(x-10))/((4-1)*(4-2)*(4-3)*(4-10)) mod 17;
DD := 8 (x + 16) (x + 15) (x + 14) (x + 7)
>E:=((x-1)*(x-2)*(x-3)*(x-4))/((10-1)*(10-2)*(10-3)*(10-4)) mod 17;
E := 8 (x + 16) (x + 15) (x + 14) (x + 13)
```

I needed to use `DD` for the fourth polynomial since `Maple` has the letter `D` used alone "reserved" for special use in differentiation.

You can check (I did!) that the Lagrange polynomials work correctly. Here is a check of the polynomial `E`.

```
>subs(x=1,E) mod 17;
0
>subs(x=2,E) mod 17;
0
>subs(x=3,E) mod 17;
0
>subs(x=4,E) mod 17;
0
>subs(x=10,E) mod 17;
1
```

And now we assemble the polynomial `P`:

```
>P:=2*A+4*B+6*C+8*DD+15*E mod 17;
P := 12 (x + 15) (x + 14) (x + 13) (x + 7)
     + 4 (x + 16) (x + 14) (x + 13) (x + 7)
     + 15 (x + 16) (x + 15) (x + 13) (x + 7)
     + 13 (x + 16) (x + 15) (x + 14) (x + 7)
     + (x + 16) (x + 15) (x + 14) (x + 13)
>PP:=expand(%) mod 17;
        3       4          2
13x + 9x  + 11x  + 9 + 11x
```

Here `Maple` was "forced" to "simplify" `P` by using the `expand` command. The result is `PP`, a version of `P` written in standard form.

I did check that `P` interpolates the correct values. Then I found its value at 0. That value can be read from the standard form already presented, of course.

```
>subs(x=1,PP) mod 17;
2
>subs(x=2,PP) mod 17;
4
>subs(x=3,PP) mod 17;
6
>subs(x=4,PP) mod 17;
8
>subs(x=10,PP) mod 17;
15
>subs(x=0,PP) mod 17;
9
```

The secret animal is the Elephant. Of course the answer would have to be either Elephant, Frog, or Banana, since those are the only animals whose numbers are between 0 and 16.

A short cut Since $P(x) = 2x$ for $x = 1$, $x = 2$, $x = 3$, and $x = 4$, there is a quicker way to create a formula for P. The polynomial E created above is 0 at 1 and 2 and 3 and 4, and $E(10) = 1$. If $x = 10$, then $2x = 20 = 3 \pmod{17}$. So the polynomial $2x + (15 - 3)(E(x))$ has the values we want of P. We need to subtract 3 from 15 in order to get the correct value for the sum of the two terms at $x = 10$. `Maple` tells me

```
>expand(2*x+12*E) mod 17;
          4     3      2
13 x + 11 x + 9 x + 11 x + 9
```

(where I used the `E` I created before). Now I can read the constant term, and it tells me again the animal is Elephant. This approach needs much less typing than the other.

Question #4 Alice and Bob are using RSA. They agree on the following modulus which is the product of two primes:

3 29220 27360 43431 10697

Alice's public key is

349 16502 12311

and Bob encrypts the number of an animal (from the table above) using her public key: he sends

84746 45152 36179 77517

What animal's name is Bob sending to Alice?

Answer Here's one method, the "classical" attack on RSA which begins by factoring the modulus.

```
>ifactor(329220273604343110697);
(45561107623) (7225905839)
```

I wrote the modulus and the other numbers in groups of 5 digits to make it easier to retype if you didn't copy the numbers with the mouse. I called the first factor `PP` and the second factor `QQ`. Of course I made the modulus not too big so that the factoring operation would not take much time. I called Alice's public key `e` and asked `Maple` to find the private key, `d`. This last computation really took very little time.

```
>e:=3491650212311;
e := 3491650212311
>msolve(e*x=1,(PP-1)*(QQ-1));
{x = 94466074575012786479}
>d:=94466074575012786479;
```

On my home computer (not fast in terms of current equipment) the factoring shown took .61 seconds of CPU time. Solving the modular equation took .04 seconds of CPU time. I checked my answer:

```
>e*d mod ((PP-1)*(QQ-1));
1
```

and then I put in the value of the message (the "ciphertext") and computed the "plaintext":

```
>mess:=84746451523617977517:
>mess&^d mod(PP*QQ);
2003
```

So the animal is the Bicycle. One of my family members complained that a bicycle wasn't an animal, but I answered that it had `two` wheels, didn't it?

I used the special `Maple` option for *fast* modular exponentiation (based on repeated squaring, as mentioned in class): `&^` instead of just `^`. This reduces the time greatly. Also, the ordinary exponentiation first computes a *huge* integer and then finds its remainder (the mod part). `Maple` may not have enough storage space for the first computation. On my machine, I will get the message `Error, object too large` indicating such an error.

Comment The web page `http://198.64.129.160/college/exam/barometr.htm` has a presentation of solutions to the well-known question: how can a barometer be used to measure the height of a building? So you don't need to do the expected.

For this problem as stated, there is an alternative method, which is true to the real cryptanalytical spirit. Breaking cryptosystems used for military or diplomatic or even business purposes frequently has not depended on methods as theoretical as those used here, but has involved such "techniques" as stealing the information, or bribing someone, or ... you get the idea. Here something even easier can be done, using what's given in the problem and the computational resources you have. Just compute N^e mod (the given modulus) for each of the 5 numbers N in the table of animals. This will certainly need less computational power than factoring the modulus, and you are guaranteed by the problem statement that you will find the answer.

Taking advantage of the type of message which is being sent is clever and has been used often. In World War II, weather stations were vital for the conduct of military observations. Many stations would send brief stylized messages such as: "The temperature is __ and the wind speed is __ from the __ direction." The cryptanalyst could then try to decrypt by guessing the likely message together with all likely keys. This approach was sometimes successful. Similarly, diplomats have on occasion helped break their own cryptosystmes by sending encrypted versions of long diplomatic communiques supplied by the "opponents"!