Fast Modular Exponentiation

Many items in public key cryptography are based on calculating modular exponents quickly. While we know we can utilize Fermat's and Euler's Theorem in certain cases to simplify calculations, for very large values of n, even these simplifications can leave an exponent that's quite large.

Consider the typical algorithm:

```
int modExp(int base, int exp, int n) {
  int ans = 1;
  for (int i=0; i<exp; i++)
    ans = (ans*base)%n;
  return ans;
}
```

This is too slow because the loop will run exp number of times, and that could be absolutely huge. (Think tens of thousands or years or more...)

Instead, we must make use of "intermediate" computations. For example, if I know that $a^{5000} = 3$ mod n, I can immediately calculate that $a^{10000} = 9$ mod n, since $3^2 = 9$ mod n. (Basically, all I am doing is squaring both sides of the first equation to yield the second equation.) This one quick step saved 5000 multiplications. Here's some sample code that utilizes this idea and is a recursive implementation of fast modular exponentiation:

```
 int fastModExp(int base, int exp, int n) {
  if (exp == 0) return 1;
  if (exp == 1) return base%n;
  if (exp%2 == 0) {
    int temp = fastModExp(base, exp/2, n);
    return (temp*temp)%n;
  }
  return (base*fastModExp(base, exp-1, n))%n;
}
```

Notice that each time the exponent is odd, it spurs a recursive call to an even exponent, and then each time a recursive call is even, it creates a recursive call to an exponent that is half in value. Thus, every 2 recursive calls result in dividing the exponent by 2. Thus, the stack depth is at most O(log exp) which is quite manageable.

The run-time of this algorithm is logarithmic in the value of the exponent, exp, instead of linear. This is a huge improvement. Consider exp = 10^{20}, $\log_2 10^{20} = 66.4$, which is much, much, much, much smaller than 100000000000000000000.

To code this iteratively, we can use the binary representation of the exponent, but this recursive version is probably the easiest to code for the average person.