Modular Exponentiation

Compute $x^n \mod m$ fast in $O(\log n)$ time

- Method 1: Recursive computation

$$x^n = \begin{cases} 1 & n = 0 \\ x^{n/2} \cdot x^{n/2} & n \text{ is even} \\ x^{n/2-1} \cdot x & n \text{ is odd} \end{cases}$$

```
int modpow( int x, int n, int m )
{
    if ( n == 0 )
        return 1 % m;

    long long u = modpow( x, n/2 m );

    if( n % 2 )
        u = ( u * x ) % m;
    else
        u = ( u * u ) % m;

    return u;
}
```

- Method 2: Iterative computation using loops

Rewrite n as its binary representation $(b_k b_{k-1} \ldots b_1 b_0)_2$. Then

$$x^n \;=\; x^{\sum_{i=0}^{k} b_i \cdot 2^i} \;=\; \prod_{i=0}^{k} x^{b_i \cdot 2^i}$$

$x^{(2^0)}, x^{(2^1)}, \ldots, x^{(2^k)}$ can be computed using a simple loop.

```
int modpow( int x, int n, int m )
{
    long long result = 1;

    while( n )
    {
        if( n % 2 )
            result = ( result * x ) % m;

        n /= 2;
        x = ( (long long)x * x ) % m;
    }

    return result % m;
}
```