2. (i) All operations are mod 101. $a = 71$. The exponents we need are 37, 18, 9, 4, 2, 1.

```
a^1 = 71
a^2 = 71*71 = 92
a^4 = 92*92 = 81
a^9 = 81*81*a = 81*81*71 = 19
a^18 = 19*19 = 58
a^37  = 58*58*a = 58*58*71 = 80
```

(ii) Assume that multiplying a k-bit number times a t-bit number gives a $(k+t)$-bit number (sometimes the sum has fewer bits). Assume that multiplying a k-bit number times a t-bit number takes kt milliseconds. $a = 1023$, so has 10 bits. The exponents we need are 1023, 511, 255, 127, 63, 31, 15, 7, 3, 1.

```
a3: a*a   (10*10 ms, product has 20 bits) *a (20*10 ms, 30 bits)
a7: a3*a3 (30*30 ms, 60 bits) * a (60*10 ms, 70 bits)
a15: a7*a7 (70*70 ms, 140 bits) * a (140*10 ms, 150 bits)
...
a1023: a511*a511 (5110*5110 ms, 10220 bits) * a (10220*10 ms, 10230 bits)
```

So the total time taken is $100((1*1+3*3+7*7+\ldots 511*511)+2*(1+3+7+\ldots 511)) = 100(347489+2*1013) = 34951500$ milliseconds.

3. (i) 1 3 27

(ii) Each time we divide y by 2, we return $x * zzz(x, y/2) * zzz(x, y/2)$. This will give us x to the power of y if and only if y is odd. So, y must be odd every time we divide it by 2. It is easy to show (e.g. by induction) that y must be exactly 1 less than a power of 2, where the smallest power of 2 is $1 = 2^0$. So, $y = 0, 1, 3, 7, 15, 31, 63$.

4. (i) There are two problems with `isp( )`. It gives the wrong answer if $n = 2$, or if $n = p^2$ where p is an odd prime (e.g. $n = 9$). So first fix the algorithm: return prime if n is 2, and change the while test to `(d*d <= n)`.

Now (with the changes above), for all integers $n \geq 2$, the algorithm is correct.

First, assume n is even. (I leave this case to you ...).

Next, assume n is odd. Notice that if n has a prime divisor, then it has a prime divisor k such that $k * k \leq n$. (Suppose $k * k > n$ and k divides n. Then $n = kj$ where $j = n/k$, and $j * j = (n/k) * (n/k) = (n * n)/k * k < n$.) So it sufficies to check among all odd numbers $\{3, 5, 7, ..., t\}$ as divisors.

(ii) Best case: the input n is even, the algorithm returns immediately, runtime $\Theta(k)$.

Worst case: the input n is prime. So there are $\Theta(\sqrt{n})$ iterations, each takes $\Theta((\lg d)^2)$ time, where d ranges from 2 to root n. So the runtime is $\Theta(\sum_{d=2}^{\sqrt{n}} (\lg d)^2)$ time.

In the sum, there are $\sqrt{n} - 1$ terms, the largest term is

$$(\lg \sqrt{n})^2 \;=\; (\lg n^{1/2})^2 \;=\; ((1/2)\lg n)^2 \;\in\; O((\lg n)^2)\,,$$

so the runtime is in $O(\sqrt{n}(\lg n)^2)$. By using integration, or by considering the last half of the terms in the sum, it is not hard to show that the sum is in $\Omega(\sqrt{n}(\lg n)^2)$. So the runtime is in $\Theta(\sqrt{n}(\lg n)^2)$.

(iii) The randomized Fermat primality test doesn't make sense when $n = 2$ or 3 (the only possible values of a would report that a is prime), so I ran it on numbers from 4 to 999.

There are no errors when the input is prime, by Fermat's little theorem. When the input is composite, the average error rate of the single-trial Fermat test is about 13 out of the first 1000 primes, so should be 0 with the 10-trial test.

```
T = [1,10]
for t in T:
  errP, errC = 0,0
  for n in range(4,1000):
    if isp(n)!=probp(n,t):
      if isp(n):  errP += 1
      else:       errC += 1
  print "errors: prime", errP, "composite", errC
```

(i) d divides a, so there exists an integer k such that $a = kd$. Similarly, there exists an integer h such that $b = hd$. Now $ax + by = kdx + hdy = (kx + hy)d$ and — since k, x, h, y are integers — $kx + hy$ is an integer, say c. So there exists an integer c such that $ax + by = cd$. So d divides $ax + by$.

(ii) By (i), there is an integer c such that $ax + by = cd$. Also, $ax + by > 0$, so neither c nor d are zero, so $d = (ax + by)/c < ax + by$.

(iii) Let $g = gcd(a, b)$. Every common divisor of a and b divides every linear combination of a and b. And, from the given formula, 51 is a linear combination of a and b, so g — a common divisor of a and b — divides 51. So $g \leq 51$.

Also, $a = 51 * 326921797$ and $b = 51 * 317907761$. So 51 is a common divisor of a and b. So $51 \leq$ the greatest common divisor of a and b, which is g. So $51 \leq g$.

So, $51 = g$.

(iv) $a = 35267 = 7 * 5038 + 1$, so 7 does not divide a. So 7 cannot be the gcd of a and b.