2.
```
def carmichael(n):
  composite = False
  for a in range(2,n):
    if 1!=gcd(a,n):
      composite = True
    elif 1!=pow(a,n-1,n):
      return False
  return composite
```

3. (i) easy to see by expansion (or use induction) (ii) upper bound: each term at most $\lg n$. lower bound: there are at least $n/2$ terms, each at least $\lg(n/2)$. (alternative solution: use an integral bound)

4. skipped

5. as n doubles, runtime increases by a factor of about 4, suggesting runtime in $\Theta(n^2)$. this makes sense if isprime(x) takes $\Theta(x)$ time. why? well, on average, how many possible factors do you need to look at before deciding whether a number is prime? and are the numbers we are using in this experiment small enough so that the time for modular division takes constant time? to answer these questions, you should put a counter in your program to answer the first question.

6.
```
def isp2(n):
  if 2==n:
    return True
  if 0==n%2:
    return False
  j=3
  while j*j <= n:
    if 0==n%j:
      return False  # composite
    j += 2
  return True      # prime
```

7. (i) d divides a, so there exists an integer t such that $dt = a$. similarly, there exists an integer s such that $ds = b$. so $ax + by = dtx + dsy = d(tx + sy)$, so d divides $ax + by$. Prove or disprove: d divides $ax + by$. (ii) $ax + by > 0$, so neither d nor $(tx + sy)$ are 0, so $d = (ax + by)/(tx + sy)$. If $d < 0$ we are done. If $d > 0$, then so is $tx + sy$, and $(ax+by)/(tx+sy) \leq (ax+by)/1 = ax+by0$. (iii) it suffices to check that 7 divides both 8616909 and 135716.

8. use extended euclid