solutions 2

2. (i) I ran this 3 times, just for fun. Your answers will be similar.

```
146.59 291.51 585.17
146.36 291.85 588.73
146.18 292.34 585.11
```

(ii) The answers will not be the same, because the values for a and b are selected by python's pseudonumber generator. If the values differ, the sums can differ.

The numbers will be similar because the variance (this is a term from statics) for this particular process happens to be small.

(iii) For the call `experiment(250)` the larger integer a is selected uniformly randomly from the interval $[n_0 = 2^{249}+1, n_1 = 2^{250}-1]$. So, using Heilbronn's formula, we expect the average number of iterations to near the interval $[.843\ln(n_0) = 145.4\ldots, .843\ln(n_1) = 146.0\ldots]$, and it is.

(iv) The runtime will be in $O(k^3)$. By assumption, there are $\Theta(k)$ iterations, and each iteration performs an integer division (to find the remainder), which can be done in $O(k^2)$ time.

The reason I do not write $\Theta(k^3)$, is because — just as with multiplication — there are faster division algorithms, so here we are just giving the upper bound, and/or assuming that division is implemented in a relatively simple way. E.g., there is an algorithm similar to Karatsuba's that has the same Θ time, only taking 2 times as long as multiplication. See the paper *Practical Integer Division with Karatsuba Complexity* by T Jebelean.

3. (i) The program recurses if and only if parameter x is at least 16.

   (ii) You should be able to trace this algorithm by hand. To check your answer, it is easy to modify the code so that it prints the answer.

```
def fastmul(x,y,depth): # x,y > 0
  for _ in range(depth):
    print '. ',
  print '(',x,y,')',bin(x),bin(y)
  n = bitlength(x)
  if n <= 4:
    return x*y
  xl, xr = leftright(x,n)
  yl, yr = leftright(y,n)
  pl = fastmul(xl,yl,depth+1)
  pr = fastmul(xr,yr,depth+1)
  pm = fastmul(xl+xr,yl+yr,depth+1)
  return (pl << 2*(n/2)) + ((pm - pl - pr) << n/2) + pr

fastmul(903,459,0)

( 903 459 ) 0b1110000111 0b111001011
.  ( 28 14 ) 0b11100 0b1110
.  .  ( 7 3 ) 0b111 0b11
.  .  ( 0 2 ) 0b0 0b10
.  .  ( 7 5 ) 0b111 0b101
.  ( 7 11 ) 0b111 0b1011
.  ( 35 25 ) 0b100011 0b11001
.  .  ( 4 3 ) 0b100 0b11
.  .  ( 3 1 ) 0b11 0b1
.  .  ( 7 4 ) 0b111 0b100
```

4. $T(n) = 2T(n/(3/2)) + \Theta(n)$. Using the master theorem, $a = 2$, $b = 3/2$, $d = 1$, and $(3/2)^1 < 2$. Also $\lg_{3/2} 2 = \lg 2/(\lg(3/2)) = 1/(\lg(3/2)) = 1/(\lg 3 - \lg 2)) = 1/(\lg 3 - 1)) = 1.7\ldots$, so $T(n) \in \Theta(n^{1.7\cdots})$.

5. (i) 953 is not a nontrivial square root of 1 mod n. But this does not help us know whether n is prime.

   (ii) 953 is a non-trivial square root of 1 mod m, so m is composite.

   (iii) 953 passes the Fermat test, so r might be prime, r might be composite.

   (iv) 953 is not a composite witness by the Miller Rabin test, so y might be prime, y might be composite. But, since y passed this test, the probability that y is composite is at most .25.

6. (a) This number n is composite.

   Proof 1: Let x =

   136325195685082786152838386460271623222828087341 65

   9938962031476239929546627487119965285916438014 2329.

   Let y =

   170746248613904456106749943543530685403391593263 91

   5102751981980693551815152243441248824724590798239 1. Notice that $n = xy$.

   Proof 2: Find some number a for which a, n fails the Fermat test. For example: 2^{n-1} (mod n) is not 1. (You have to check this.) So, by Fermat's theorem, n cannot be prime.

   Proof 3: run the MillerRabin test, and you will soon discover a base that fails the Fermat test (like I did in the previous example), or a non-trivial square root of 1 mod n.

   (b) This number n is probably prime. Evidence will be any randomly selected number or set of numbers (the more numbers, the more evidence) in the range `[2,n-2]` for which the MillerRabin test does not find a composite witness. So, I could give you my list of say 10 numbers, and you could run MillerRabin on them, and confirm that each number is not a composite witness.

   Weaker evidence is to just run the Fermat test on randomly selected in `[2,n-2]`. Just for fun, I checked: as expected, for every integer t in `[t,10000]`, t^{n-1} mod n = 1.

7. $n/(\lg n) \in \Theta(n/(\log n)$ $\quad n + 4\lg^4 n \in \Theta(n)$ $\quad 6n + 9n^2 + n^1.5 \in \Theta(n^2)$

   $10^{\lg n} \in \Theta(n^{\lg 10})$ $\quad n/(\lg(3n)) \in \Theta(n/(\log n)$ $\quad 2^{\log_3 n} \in \Theta(n^{\log_3 2})$

   $n^{\log_3 2} \in \Theta(n^{\log_3 2})$ $\quad n^{\lg_2 3} \in \Theta(n^{\lg 3})$ $\quad n^{1.6} \in \Theta(n^{1.6})$

   $\Theta(n^{\log_3 2})$ $\quad \Theta(n/\log n)$ $\quad \Theta(n)$ $\quad \Theta(n^{\lg 3})$ $\quad \Theta(n^{1.6})$ $\quad \Theta(n^2)$ $\quad \Theta(n^{\lg 10})$

8. (i) $a = 2, b = 2, d = 1.5$ and $b^d 2^{1.5} > 2 = a$, so $T(n) \in \Theta(n^{2^{1.5}}) = \Theta(n^{2.82\ldots})$.

   (ii) subproblems of different sizes, so does not apply

   (iii) $a = 3, b = 2, d = 1.5$, and $b^d = 2^{1.5} < 3 = a$ so $T(n) \in \Omega(n^3)$

   (iv) $a = 16, b = 4, d = 2$ and $b^d = a$so $T(n) \in O(n^2 \log n)$

9.
```
integer  1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20
inverse  1 11  * 16 17  *  *  8  *  19  2  * 13  *  *  4  5  * 10 20
```

   (ii) 104729 * 369 + 1000 * -38645 = 1, so a inverse = -38645 = 66084 (mod n)

   (iii) The gcd of a and n is 7, which is not 1, so a has no inverse mod n.