last name _____ **id#**_____

cmput 204 start of assignment 2 due start of class, Oct 5

1. **If you leave this question blank, your assignment will not be marked and its weight will be transferred to the final exam.**

   Acknowledge **all** sources, including all references and all people with whom you discussed any part of any question (for each discussion, list the relevant questions):

2.
```
import random
def gcd(a,b):
  iterations = 0
  while b>0:
    iterations += 1
    a, b = b, a % b
  return a, iterations
def experiment(bits):
  n, experiments, sum = pow(2,bits), 100, 0
  for j in range(experiments):
    a = random.randint(n/2+1,n-1)
    b = random.randint(n/4+1,n/2-1)
    sum += gcd(a,b)[1]
  print sum/(1.0*experiments)
experiment(250); experiment(500); experiment(1000)
```

   (i) Run this on your computer. Give the output.

   (ii) Will your answer to (i) be the same as your classmate's? Similar? Explain briefly.

   (iii) Heilbronn showed that the average number of iterations in Euclid's gcd algorithm is about $.843 \ln n$, where n is the larger of the two inputs. How does this compare with your output?

   (iv) Assume that the average number of iterations in gcd(n,b) is $\Theta(k)$, where $k \in \Theta(\log n)$. Using Θ notation, as a function of k, give the average runtime of gcd(n,b). Justify briefly.

3. Here is a recursion tree diagram for Karatsuba's `fastmul(95,99)` from the webnotes, with binary representations written after each call.

```
( 95 99 ) 0b1011111 0b1100011
.  ( 11 12 ) 0b1011 0b1100
.  ( 7 3 ) 0b111 0b11
.  ( 18 15 ) 0b10010 0b1111
.  .  ( 4 3 ) 0b100 0b11
.  .  ( 2 3 ) 0b10 0b11
.  .  ( 6 6 ) 0b110 0b110
```

(i) Explain why `(11,12)` does not generate further calls, but `(18,15)` does.

(ii) Give the recursion tree diagram, also with binary representations, for `(903,459)`.

4. Abustarak's recursive fast multiplication algorithm for two n-bit numbers requires 97 n-bit addition/subtractions, but only 2 recursive multiplications on $(2/3)n$-bit numbers. Give the runtime of Abustarak's algorithm. Recall: $\log_b c = (\lg c)/(\lg b)$.

5. You want to know whether integers $n, m, r, y > 1000$ are prime. For each case, what can you conclude? Justify briefly.

   (i) The python command `pow(953,2,n)` returns 2.

   (ii) The python command `pow(953,2,m)` returns 1.

   (iii) The python command `pow(953,r-1,r)` returns 1.

   (iv) The webnotes python function `compositeWitness(953,y,True)` returns False.

6. (a) For the 199-digit number n below, using code from the course webnotes, either

   (i) give a proof (that anyone who has access to python can check) that n is composite, or

   (ii) give evidence that n is prime. Explain briefly.

   23277015754784320415680511129531323642539596301079
   25231126314751166886247680031277741025955350572733
   72882035061374185329962907157908214243547046441078
   19730894655385487355395868736069022772698057286 39

   (b) Repeat (a) for this number:

   44551500502410842113804182063960938859653291027851
   62045912774353570323307393258540221313160825849347
   15976774424570113107151092355600238332977845528872
   92168870424093307967494114938291081167162997372 73

7. (i) For each of the following functions $f(n)$, give the simplest function $g(n)$ such that $f(n) \in \Theta(g(n))$. Recall: $\lg n$ is log base 2 of n; $\lg^t n = (\lg n)^t$; $a^{\log_b c} = c^{\log_b a}$.

   $n/(\lg n)$ $\quad n + 4\lg^4 n$ $\quad 6n + 9n^2 + n^1.5$ $\quad 10^{\lg n}$ $\quad n/(\lg(3n))$ $\quad 2^{\lg_3 n}$ $\quad n^{\lg_3 2}$ $\quad n^{\lg_2 3}$ $\quad n^{1.6}$

   (ii) Sort the resulting functions $g(n)$ from (i) in increasing order of complexity.

8. For each of the following, use the master theorem to give the O or Θ or Ω runtime, whichever is best, or explain why the master theorem does not apply. In each case, assume that $T(n)$ is some positive constant for small values of n.

   (i) $T(n) = 87n \times \sqrt{n} + T(n/2) + T(n/2)$

   (ii) $T(n) = T(n/2) + T(n/4) + 6n$

   (iii) $T(n) \geq n^{1.5} + 3T(n/2)$

   (iv) $T(n) = O(n^2) + 16T(n/4)$

9. (i) For each integer, give its inverse mod 21, or write `*` if the inverse does not exist.

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

   inverse
   ```

   (ii) Let $a = 1000$ and $n = 104729$. Run `exteuclid(n,a)` from webnotes. Give the last line of output. Give the inverse of a mod n, or explain why it does not exist.

   (iii) Repeat (ii) with $a = 35833$ and $n = 682843$.