Math 55 - Spring 2004 - Lecture notes # 7 - February 10 (Tuesday)

Goals for today: continue integer algorithms from last time:
division algorithm:
hash tables
generating random numbers
gcd (greatest common divisor)
Euclidean algorithm
We will not cover section 2.7 of text (subject of Ma54)

Anyone still on waiting list has to see Michael West! (still 6 people?)

In class Midterm next Tuesday!
See on-line Course Outline
(no-make ups, use higher of Midterm, Final grades)
Will cover through section 2.3 (homework due this week),
plus one definition from today (mod)
Will bear similarities to earlier midterms (see web page)
No Cheat sheets!

Homework for next time (light):
(1) What is the prime factorization of 15!
(2) How many zeros are at the end of 15!,
when written as a decimal number?
(3) Find integers s and t such that s*22 + t*36 = gcd(22,36).
(4) What is gcd(210,429)? lcm(210,429)?

Application of Division Algorithm:
Computing the gcd(a,b) using the Euclidean algorithm
(much cheaper than factoring a and b into prime factors)

```
% assume a and b nonnegative, at least one nonzero
x = a
y = b
while y != 0
  r = x mod y
  x = y
  y = r
end while
return(x)
```

EX: gcd(14,10)

```
x = a = 14, y = b = 10
Loop 1: r=14-1*10=4;   Loop 2: r=10-2*4=2;   Loop 3: r=4-2*2=0;
        x=10;                  x=4;                  x=2;
        y=4;                   y=2;                  y=0;  return(2)

Proof of correctness of algorithm:
 two things to prove: 1) that it terminates in a finite
                         number of steps and
                      2) that it returns the right answer
  Proof of 1):
     After each pass through the while-loop, x and y get replaced
       by new values.
     In particular, y gets replaced by r=x mod y
     By the definition of mod, 0 <= r < y, so the new value of y is
       strictly less than the old value of y, and at least 0
     Since y keeps decreasing, and is >= 0, it must eventually hit 0,
       at which point the while loop stops
ASK&WAITS:  What is an upper bound on the number of time we go around the loop?

  Proof of 2):
    We will do this in three steps:
    1) we will show that gcd(x,y) at the end of the loop is the same
       as gcd(x,y) at the end of the loop, after x and y are updated
    2) when we finally exit the loop, (x,y) = (x,0), and so
       gcd(x,y) = gcd(x,0) = x, which is what we return
    3) Therefore gcd(a,b)  = gcd(x,y) before start of loop
                           = gcd(x,y) after one pass through loop
                           = gcd(x,y) after two passes through loop
                           = ...
                           = gcd(x,y) after last pass through loop
                           = gcd(x,0) since y=0 when loop terminates
                           = x, which is what we return

   To finish proof, need to prove step 1):
   Lemma: gcd(x,y) = gcd(y, x mod y)
   Proof: Let r = x mod y, so x = q*y+r, 0 <= r < y.  We will show that
          d|x and d|y if and only if d|y and d|r. Thus x and y
          have the same set of common divisors as y and r.
          In particular they must have the same greatest common divisor.
          First suppose d|x and d|y, then we have to show d|y and d|r.
             d|y is easy, and since r=x-q*y, d|r too.
          Second suppose d|y and d|r, then we have to show d|x and d|y.
```

```
                    d|y is easy, and since x = q*y+r, d|x too.

   Note: Since gcd(x,y) stays the same after each pass through the loop, we
         call gcd(x,y) a "loop invariant". Finding a loop invariant is a
         common proof technique for proving programs compute the right answer.

  EX: Find integers s and t so that s*10 + t*14 = gcd(10,14) = 2
      More generally, we can always find s and t so that s*a + t*b = gcd(a,b)

ASK&WAIT: Guess s and t

      More systematically, work forwards through Euclidean algorithm,
         finding integers ax and bx so x = ax*a + bx*b
         and     integers ay and by so y = ay*a + by*b
         at the end of each loop iteration

      x = a = 14; y = b = 10;
      Loop 1: r=14-1*10=4;     Loop 2: r=10-2*4=2;   Loop 3: r=4-2*2=0;
              x=10;                    x=4;                  x=2;
              y=4;                     y=2;                  y=0;  return(2)
      Start of Loop 1: x = a = 1*a + 0*b  y = b = 0*a + 1*b
                             = ax*a+bx*b        =ay*a +by*b
      End of Loop 1:   x = y = 0*a + 1*b  y = r = x-1*y = (1*a+0*b)-1*(0*a+1*b)
                                                        = 1*a -1*b
                             = ax*a+bx*b                =ay*a +by*b
      Start of Loop 2: x and y are same as at end of Loop 1
      End of Loop 2:   x = 1*a -1*b       y = r = x-2*y = (0*a+1*b)-2*(1*a-1*b)
                                                        = -2*a+3*b
                       = ax*a+bx*b                      = ay*a +by*b
      Start of Loop 3: x and y are same as at end of Loop 3
      End of Loop 3:   x = -2*a+3*b       y = 0
                       = ax*a+bx*b
      Finally, s=ax=-2 and t=bx=3 satisfy gcd(a,b)=s*a+t*b as desired.

ASK&WAIT:  If x = ax*a + bx*b and y = ay*a + by*b at the beginning of the
           loop body, what are they at the end?
           Notation: use x,y,ax,bx,ay,by to mean values at start of loop,
                     x',y',ax',bx',ay',by' to mean values at end of loop,

 Fact: cost of algorithm is O(log min(x,y)) (proof in Chapter 3)
       much less than factoring!
 Will use gcd later again.
```