Some Basic Ideas you learned in grade school

Given 2 positive integers \( x, y \)

1) Greatest Common Divisor (gcd)
   is the largest positive integer that divides both \( x \) and \( y \).

\[ \text{Ex: } \gcd(15, 6) = 3, \quad \gcd(28, 14) = 14 \]

2) Least Common Multiple (lcm)
   is the smallest positive integer which is divisible by both \( x \) and \( y \).

\[ \text{Ex: } \text{lcm}(15, 6) = 30 \quad \text{since} \quad 30/15 = 0 \quad (\text{Rem} \ 0) \\
\text{30/6 = 5 (Rem} \ 0) \quad \text{and No Smaller value works} \quad \text{Try It!} \]

\[ \text{Ex: } \text{lcm}(28, 14) = 28 \quad \text{since} \quad 28/28 = 1 \quad \text{and} \quad 28/14 = 2 \quad (\text{Certainly, No Integer < 28 Divides 28}) \]

Fact: Usually harder to compute lcm than gcd.

Fact: For any positive integers \( m, n \)

\[ \text{lcm}(x, y) \times \gcd(x, y) = x \times y \]  

(i.e., if you know \( \gcd \), can compute \( \text{lcm} \) easily)
Euclid's Algorithm to Compute $\gcd(x, y)$:

1. Without loss of generality, assume $x > y$.
2. Define the sequence of division algorithm iterations.

\[
\begin{align*}
(1) \quad & x = q_1 y + r_1 \rightarrow \frac{x}{y} = q_1 \text{ Remainder } r_1 \\
(2) \quad & y = q_2 r_1 + r_2 \rightarrow \frac{y}{r_1} = q_2 \text{ Remainder } r_2 \\
(3) \quad & r_1 = q_3 r_2 + r_3 \rightarrow \frac{r_1}{r_2} = q_3 \text{ Remainder } r_3 \\
(4) \quad & r_2 = q_4 r_3 + r_4 \rightarrow \frac{r_2}{r_3} = q_4 \text{ Remainder } r_4 \\
& \quad \vdots \\
(N-1) \quad & r_{N-2} = q_{N-1} r_{N-1} + r_N \rightarrow \frac{r_{N-2}}{r_{N-3}} = q_{N-1} \text{ Remainder } r_N \\
(N) \quad & r_{N-1} = q_N r_N
\end{align*}
\]

Where $r_N$ is the last non-zero remainder.

Then, $\gcd(x, y) = r_N$. 

Euclid's Algorithm in Words:

(1) \( \frac{\text{Dividend} \ (x)}{\text{Divisor} \ (y)} \rightarrow \text{Quotient} \ (q), \text{remainder} \ (r) \)

(2) If Remainder = 0, STOP (\( \gcd(x, y) = \text{last non-zero remainder} \))

(3) Toss Quotient
(4) Move Divisor to Dividend
(5) Move Remainder to Divisor
(6) Start at (1) again.

Ex: \( \gcd(87, 18) \)

(1) \( \frac{87}{18} = 4 \text{ Rem } 15 \)

(2) \( \frac{18}{15} = 1 \text{ Rem } 3 \)

(3) \( \frac{15}{3} = 5 \text{ Rem } 0 \)

The last non-zero remainder was 3
Hence, \( \gcd(87, 18) = 3 \).
Questions

• Does Euclid's Alg always work
  → yes

• How fast does it converge to $\gcd(x, y)$
  → Every other iteration, the value of dividend reduced by a factor of at least 2. (Converges Exponentially fast)
    → That's fast}
Euclid's Algorithm can be broken down into several steps - essential to create computer algorithm:

1. Declare Variables & Initialize

\[
\text{Dividend } x \\
\text{Divisor } y \begin{cases} \text{Need to initialize these} \\ \text{(Input Data)} \end{cases}
\]

\[
\text{Quotient } q \begin{cases} \text{These are computed values} \\ \text{(Output Data)} \end{cases}
\]

\[
\text{Remainder } r
\]

2. Divide \( x \) by \( y \), toss quotient, giving remainder.

\[ \text{Modulo Division operation} \]

(Will Discuss Monday)

In C, C++, Java, represented by

\[ r = x \% y; \]

\[ r \equiv x \mod y \]

3. Test to see if \( r > 0 \)

(If it is not, we're done - write \( \gcd(x, y) \) as the best value of \( x \) computed)

4. Move Divisor to Dividend

\[ x = y; \]

5. Move Remainder to Divisor

\[ y = r; \]

6. Go back to Step 2.
A Simple Java Applet to do Euclid's Algorithm

→ Go to .../list230/labs/lab64/GCD-Simple directory, Download

GCD.java
GCD.html

→ into C:\Temp\Temporary\Internet\Files
(Don't seem to have much choice here
unless you have a floppy)

→ Now create a new directory
C:\Temp\GCD
and copy these 2 files into this directory.

→ Start up any text editor/Java IDE
you like
(I like JPad, in the Programming
Subdirectory of Start Menu/Programs;
You can compile and test your
applet using the Sun JDK within
JPad. It's very nice and Simple)
Note that the program GCD.java is not completed:

→ The entire program is embedded in a single class GCD (which extends the Applet class)

The methods

init(), action(), takeStep()

deal strictly with initializing the applet and x, y values, and provide a rudimentary graphic interface

→ We don't touch these.

The method
gcd()
Contains all the computations

→ Note the variables named

x1, y1, r1
(Why don't we need q1 ?)
\text{OUTPUT is an Integer}
\text{x, y, are Integers}

\text{int } \gcd (\text{int } x, \text{ int } y, ) \{ \\
\text{int } r_1 \quad \text{// Declare remainder as integer}
\text{while } (r_1 > 0) \{ \\
\quad \text{--- you fill in code here ---} \\
\quad \} \\
\text{return } x_1 \quad \text{// Send back last}
\} \\

\text{The while loop causes the code you fill in here to continue to execute until } r_1 > 0