Nov 23rd 2016 CE

Recursion

CP213 Lecture Start

Lisp (1960) → first program language (and used Recursion)

function review

```
public static int F(int x, int y) {
    int a, b;
    a = x * x - y;
    if (a > 0) { b = g(a, x, y); }
    else { b = ... }
    return b; }
public static int g(int a, int x, int g) { int a, b ... return ...; }
public static void WV (int x) {
    Integer y = new Integer(x);
    String S = y.string;
    for(int i=0; i < s.length; i++)
        System.out.println(S.charAt(i));
}
```

// non recursive ↑ recursive ↓ (& using only Int ↓ arithmetic)

X % 10 → gives last digit X / 10 → gives last digit +

```
public static void WV (int x) {
    if(x > 0) { WV(x/10); }
    System.out.println(x%10); }
```

CP213 Lecture Start Nov 25th 2016 CE

Euclid Algorithm → Find gcd of two integers

$$gcd(a,b) = \begin{cases} a, & a=b \\ gcd(a\%b, b), & \end{cases}$$

Ch 13 Recursion

x^n $(n \in \mathbb{Z} \geq 0)$ $x^n = x \cdot x^{n-1}$

```
double pow_n (double x, int n) {
    double f = 1.0;
    for (int i = 1; i <= n; i++) {
        f = f * x; }
    return f; }
```

Recursive

```
double pow_n (double x, int n)
    if (n = 0) {
        return 1; }
    else { return (x * pow_n(x, n-1)); }
```

or $x^n \equiv (x^{\frac{n}{2}})^2$ n is even

$x^n \equiv (x^{\frac{n-1}{2}})^2 \cdot x$ n is odd

```
double pow_n (double x, int n)
    if (n = 0) { return 1; }
    else {
        double temp = pow_n(x, n/2);
        if (n % 2 == 1) { return temp * temp * x; }
        else return { temp * temp; }
    }
```