Recursion

- A recursive function is one that calls itself
  - remarkable concept in Computer Science

- General form:
  - base case
    - solution for a trivial case
    - it is also used to stop the recursion in algorithms, preventing “stack overflow”
    - every recursive algorithm needs at least one base case
  - recursive call(s)
    - divide problem into smaller instances!
    - solve smaller instances with the same code
**Example: factorial**

**Definition.** The factorial function is defined by the product:

\[ n! = \prod_{k=1}^{n} k \]

or by the recurrence relation:

\[ n! = \begin{cases} 1 & \text{if } n = 0 \\ (n-1)! \times n & \text{if } n > 0 \end{cases} \]

```c
1 int factorial(int n) {
2     // base case
3     if (n < 2) {
4         return 1;
5     }
6     // recursive call
7     return n * factorial(n-1);
8 }
```

**Example: power of a number**

**Definition.** The \( n \)-th power of \( b \) is defined by:

\[ b^n = \underbrace{b \times b \times \cdots \times b}_{n \text{ times}} \]

```c
1 int power(int x, int n) {
2     // base case
3     if (n == 0) {
4         return 1;
5     }
6     // recursive call
7     return x * power(x, n-1);
8 }
```

**Example: sum of array**

```c
1 int sum_array(int *A, int n) {
2     if (n == 1) {
3         return A[0];
4     }
5     // recursive call
6     return A[n-1] + sum_array(A, n-1);
7 }
```
Example: decimal to binary

```c++
1 void print_binary(int n) {
2     if (n > 0) {
3         print_binary(n/2);
4         std::cout << n % 2;
5     }
6 }
```

Example: indirect recursion

```c++
1 void f2(int n);
2
3 void f1(int n) {
4     if (n > 1) {
5         std::cout << "1";
6         f2(n - 1);
7     }
8 }
9
10 void f2(int n) {
11     std::cout << "0";
12     f1(n - 1);
13 }
```

Review

- How programs are executed?
- Memory model
  - stack and heap
- How parameters are passed to functions?
  - passing arrays
- What are pointers?