CSC 212: Data Structures and Abstractions
Recursion

Marco Alvarez
Department of Computer Science and Statistics
University of Rhode Island

Spring 2018

THINK BIG - WE DO

Administrativia

- Gradescope
  - writing ‘correct’ code is not easy (especially in C/C++)
  - requires practice, discipline, perseverance, resilience

- Piazza
  - test cases can be shared, however, in an organized way
  - can also be used to discuss lecture topics

Announcements

- Weekly Assignment Videos
  - instructor creates a special Note (during TA meeting)
  - youtube links should be posted on Piazza as followup comments (1 link per group)

- Special Topics Videos
  - instructor posts a list of suggested topics on Piazza (during TA meeting)
  - youtube links should be posted as followup comments under the same Note

Functions

```c
1 int foo(int n) {
2   int a = 10;
3   n += a;
4   return n;
5 }

6 int bar(int n) {
7   int a = 10;
8   n = foo(n);
9   return a * n;
10 }
```

bar(5) ?
Recursion

- Solve a task by reducing it to smaller tasks (of the same form)
  - powerful tool in computer science
- Technically, a recursive function is one that calls itself
- 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 using the same code

Demo

- Count the number of students behind you
  - … and include yourself

General Form

```javascript
function() {
  if (this is the base case) {
    calculate trivial solution
  } else {
    break task into subtasks
    solve each task recursively
    merge solutions if necessary
  }
}
```
**Example: factorial**

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

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

or by the recurrence relation:

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

---

**Day in a life of a recursive call**

```c
int val = factorial(4);
```

```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 }
```
Precondition

- Specify the **input conditions** to a function (contract)
  - what is the expected input for the factorial example?
- Use **assert**
  - terminates program execution
- Use **throw, try, catch**
  - throws an exception that can be caught later
- . . . or just let the program misbehave or crash later

Example: power of a number

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

\[
 b^n = b \times b \times \ldots \times b \quad \text{\( n \) times}
\]

```java
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 }
```

Indirect Recursion

```java
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 }
```

Need a Break?

https://www.youtube.com/watch?v=s0Qh-gy7ktA

https://www.youtube.com/watch?v=ChEPRJ4edUk
Example: sum of array

```c
1 int sum_array(int *A, int n) {
2   if (n == 1) {
3       return A[0];
4   }
5   return A[n-1] + sum_array(A, n-1);
6 }
```

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 }
```

Binary Search

```
1 2 5 10 15 20 22 30 35 40 43 48 51
```

low

k = 48?

high

Binary Search

```
1 2 5 10 15 20 22 30 35 40 43 48 51
```

low

mid

k = 48?

high
Binary Search

1 2 5 10 15 20 22 30 35 40 43 48 51

low high

k = 48?

Binary Search

1 2 5 10 15 20 22 30 35 40 43 48 51

low mid high

k = 48?

Binary Search

1 2 5 10 15 20 22 30 35 40 43 48 51

low mid high

k = 48?

Binary Search

1 2 5 10 15 20 22 30 35 40 43 48 51

low high

k = 48?

k = 22?

k = 0?

k = 51?

k = 55?

n = 13
The version of binary search that I wrote for the JDK (java.util.Arrays) contained the same bug. It was reported to Sun recently when it broke someone’s program, after lying in wait for nine years or so.

Extra, Extra - Read All About It: Nearly All Binary Searches and Mergesorts are Broken
Friday, June 02, 2006
Posted by Joshua Bloch, Software Engineer

```java
int mid = (low + high) / 2;
```

```java
int mid = lo + ((hi-lo) / 2);
```