Announcements

- Homework 1 has been posted
- Outline
  - Runtime
  - Recursive programming
  - ADT
  - Lists

- Reading:
  - Runtime stuff in Chapter 2
  - Chapter 3.1,3.2
Example 9

```c
int Example9( int n ) {
    if( n <= 1 ) {
        return 1;
    }
    else {
        return Example9(n -1) + Example9(n-2);
    }
}
```
Fibonacci

Anyone know what the Fibonacci series is?
In nature
Recursion

- Fib is related to recursion
- Recursion is code that is defined in terms of itself
- Many DS problems can be solved in a recursive fashion
Example

- We want to code a method Power(x, y) which raises x to the power of y

- Can you code the power function as a recursive method?
int power(int x, int y) {
    if(y==0) {
        return 1;
    }
    else {
        return x * power(x,y-1);
    }
}
- What did we forget??

- Negative examples
  - Add in check for negative
some important points

- Biggest problem with recursive code is getting stuck in infinite loop

- Here are some quick guidelines on how to write recursive methods
Rules

1. Base Case:
   make sure you have a base case which can be computed without recursion

2. Progress
   make sure you are making progress towards solution
More Rules

3. Assume all recursive calls return correctly
   1. When you make a sub call with yourself assume the right answer is returned (base case is reached)

4. Never duplicate work
   1. Is you can try to have the answer somewhere
Question

- when is recursion necessary to solve a problem ??
Never

if you can do it recursively, can do it iteratively

can you prove this ??
Proof

- Your cpu is not a recursive cpu
  - Even if it’s a mac 😊

- generally a non recursive solution will run faster

- BUT
  - harder to read
Example

- count number of digits in an int recursively

```c
int numDigits(int x) {
    if(abs(x) < 10) {
        return 1;
    }
    else {
        return numDigits(x/10) + 1;
    }
}
```
tail recursion

- Most of the time, when you see recursion, it's slowed down by the fact that it needs to wait for another function call before returning its value.

- Example on board
Factorial

int factorial(int num) {
    if (num == 1)
        return 1;
    else
        return num * factorial(num-1);
}

Can you program it so it wont create a tail?
int fact2(int num, int result) {
    if (num == 1)
        return result;
    else
        return fact2(num - 1, result * num);
}
**switch gears**

- let us start to talk about how to organize data
Abstraction

- one important concept for DS is the idea of abstraction
- anyone have a pilots license ??
- anyone know how to fly a plane ??
Abstraction is hiding the details

- focus on important bits
- simplify
- allow change to happen later
  - we can replace underlying structure without changing the outside view
  - same idea of a standard api
List ADT

- We want to represent a group of items

- with a list which operations would you have ??
List ADT

- insert
- remove
- sort

- find first
- find last
- count

Notice how we aren’t even talking about how to store the list

Any IDEAS ??
Arrays to implement List ADT

- Positive ?
- Negative ?

- what is the cost of insertion ?
- what is the cost of insert at beginning ?
- find by value ?
- find by index ?
Linked List

- structure which has list and links to elements
- each node points to next
- not necessarily adjacent in memory
- last has null pointer
- need to keep link to first element
Linked Lists

- Positive ?
- Negative ?

- what is the cost of insertion ?
- what is the cost of insert at beginning ?
- find by value ?
- find by index ?
improvements

- can improve linked list DS by adding another set of links going backwards
- Double linked lists
- header / tail nodes