Recursion

Good examples:

* max/min/average
* towers of hanoi

**Day 1 (has slides)**

* Warmup with students writing (iterative) fact by hand.
* Show answer in C++, run a few times.
* Show relationship between different calls to fact (on slides).
* Introduce idea of recursion (on slides)
* Introduce base case/recursive case.
  + Go through BC/RC for factorial.
  + BC: if n=1, fact(n) = 1.  
    RC: if n>=2, fact(n) = fact(n-1)\*n USE LOCAL VARIABLE.
* Write Java version.
* Do memory diagram for factrec(3).
* Demo turtle graphics.
* Do uc(n)
  + other alternatives – add up all numbers in an array.
  + find max number in an array.
  + detect if palindrome
  + detect if lowercase
  + **reverse a string**
  + count letters in string

NOTES FROM S17: barely got through uppercase(s) example. Suggest do more quickly iterative fact that students warmup with --- put on board, let them do at start.

**Day 2**

* Start by showing uc formulation written out.
  + uc(‘c’) = toupper(‘c’)
  + uc(“bc”) = toupper(‘b’) + toupper(‘c’)
  + uc(“abc”) = toupper(‘a’) + toupper(‘b’) + toupper(‘c’)  
      
    OR
  + uc(‘a’) = toupper(‘a’)
  + uc(“ab”) = toupper(‘a’) + toupper(‘b’)
  + uc(“abc”) = toupper(‘a’) + toupper(‘b’) + toupper(‘c’)
* **Try this in a bunch of ways:**
* **void** weird(**int** n)  
  {  
   **if** (n == 0)  
   **return**;  
   **else** {  
   weird(n - 1); // optional  
   cout << n << endl;  
   weird(n - 1); // optional  
   }  
  }

Day2 record from s17

* quick review of factorial
* show Uppercase
* show how they are similar
* show debugger
* Do three versions of the weird function – showing call diagrams/mem diagrams each time.
  + one with print before cout
  + one with print after cout
  + one with prints before and after
* Do towers of Hanoi – this is kind of fast and rushed. Needs at least ten minutes to do properly.

Day 3

* Start with Hanoi.
  + Go over recursive formulation.
  + Show example (python code in folder).
  + Ask students to take note of WHICH disk is being moved each time. (1 2 1 3 1 2 1...)
    - Same as weird.
  + Then uncomment line in python code to show longer example.
* Lab time (see day3 lab docx).
  + Go over POWER at end.
  + formulation 1:
    - power(x, y) = x \* power(x, y-1)
    - But this is slow...does n multiplications.
    - Better: power(x, y) =
      * if y is even: power(x, y/2) \* power(x, y/2).
      * If y is odd: x \* power(x, y/2) \* power(x, y/2).

Day 4 – binary search

* First, illustrate with tearing phone book in half.
* Then, show example of how we would do this with a vector of ints in C++.
  + go through example with numbers on slides. No code or pseudocode yet.
  + OPT: Have students call out indices of middle point as we do this.
* Introduce pseudocode....with handout on slides and physical hard copies here.
* Do slide example again, showing values of variables. low, high, mid
* Students write code in pairs (there is starter code).
* Talk about big oh (logarithmic).
  + For a one-item length list, how many calls to binsearch are there?
    - 1
  + Now if you have 2 items in list, how many calls are there.
    - 2
  + For 4 items,
    - 3 calls.
  + For 8 items, 4 calls.
  + For 16 item, 5 calls.
* What is the general formula?
  + note that 2^0 = 1
  + 2 ^ 1 = 2
  + 2 ^ 2 = 4
  + 2 ^ 3 = 8
  + 2 ^ 4 = 16
* So general formula for array of size n is log\_2(n) + 1.