Recursion

Recursion: If you get the point, stop; otherwise, see Recursion.

Infinite recursion: See Infinite recursion.

Read: pp. 403-408 but SKIP sect. 15.1.2

Look in ProgramLive CD, page 15-3, for some interesting recursive methods.

Download presented algorithms from the website

Recursive definition: A definition that is defined in terms of itself.

Recursive method: a method that calls itself (directly or indirectly).

Recursion is often a good alternative to iteration (loops), which we cover later. Recursion is an important programming tool. Functional languages have no loops—only recursion.

/** = the number of ‘e’s in s */
public String noe(String s) {
  if (s.length() == 0) {
    return 0;
  } // { s has at least one char
  if (s.charAt(0) != 'e') {
    // return number of e’s in s[1..];
    return noe(s.substring(1));
  } // { first char of s is ‘e’
  // return 1 + (number of e’s in s[1..])
  return 1 + noe(s.substring(1));
}

Factorial: !0 = 1
!n = n * !(n-1) for n > 0

Step 1: HAVE A PRECISE SPECIFICATION
public static int fact(int n) {
  if (n == 0) {
    return 1;
  } // {n > 0
  return n * fact(n-1);
}

Step 2: Check the base case.
When n = 0, 1 is returned, which is 0! So the base case is handled correctly.

Step 3: Recursive calls make progress toward termination.
/** = !n (for n>=0) */
public static int fact(int n) {
  if (n == 0) {
    return 1;
  } // {n > 0
  return n * fact(n-1);  // recursive case
}

Step 4: Recursive case is correct.

Factorial: !0 = 1
!n = n * !(n-1) for n > 0

Task: Write a method that removes blanks from a String.

0. Specification:
/** = s but with its blanks removed */
public static String deblank(String s)

1. Base case: the smallest String is “”.
   if (s.length() == 0) {
     return “”;
   }

2. Other cases: String s has at least 1 character.
If it’s blank, return s[1..] but with its blanks removed.
If it’s not blank, return s[0] + (s[1..] but with its blanks removed)
public static String deblank(String s) {
    if (s.length() == 0)
        return s;
    // {s is not empty}
    if (s[0] is a blank)
        return s[1..] with its blanks removed
    // {s is not empty and s[0] is not a blank}
    return s[0] + (s[1..] with its blanks removed);
}

The tasks given by the two English, blue expressions are similar to the task fulfilled by this function, but on a smaller String! Rewrite each as deblank(s[1..]) .

Creating a recursive method
Notation:
s[i] shorthand for s.charAt[i].
s[i..] shorthand for s.substring(i).

public static boolean isPal(String s) {
    if (s.length() <= 1)
        return true;
    // { s has at least two characters }
    return s.charAt(0) == s.charAt(s.length()-1)  &&
        isPal(s.substring(1, s.length()-1));
}

A man, a plan, a careet, a ban, a myriad, a rare, a liar, a hoop, a pier, a cataupa, a gas, an oil, a bird, a yel, a vat, a cow, a pas, a wag, a tax, a ray, a rum, a cap, a yum, a gnu, a bar, a wall, a car, a laguer, a ward, a bin, a woman, a vassal, a wolf, a tume, a nit, a pull, a first, a wait, a boy, a dash, a tan, a cab, a datum, a gall, a hat, a fup, a zap, a soy, a jaw, a lay, a wet, a gapple, a tug, a trot, a trap, a team, a torn, a caper, a top, a took, a toll, a ball, a fair, a sun, a minum, a tinner, a passer, a capital, a rut, a mines, a treed, a cabul, a tangle, a rum, an ass, a maw, a zap, a jam, a dam, a sub, a salt, an atom, a sal, an ad, a wushi, a radian, a room, a road, a zip, a lad, a parish, a revel, a real, a old, a post, a play, a pig, a peek, a parabola, a dog, a pat, a rad, a nu, a fan, a pul, a ren, a nod, an esa, a lay, an ec, a battle, a mug, a mot, a map, a maxim, a woold, a leer, a grub, a gub, a gel, a dub, a cradel, a total, a orl, a tap, a gag, a rat, a maner, a bur, a gal, a cola, a pap, a yaw, a taj, a raj, a gab, a mug, a pagan, a bug, a jar, a but, a way, a pepu, a locul, a gar, a thorn, a mat, a rug, a gap, a tar, a decal, a tot, a fell, a tue, a band, a tip, a boy, a know, a docom, a mix, a map, an atom, a gur, a kil, a bulken, a gale, a ten, a don, a morla, a pan, a fam, a ducat, a pagoda, a loh, a rap, a keep, a nip, a gimp, a loop, a dozen, a lees, a horn, a pod, a tap, a dial, a more, an aid, a raud, a wad, an alias, an arr, an atlas, a bus, a mudan, a jag, a saw, a mors, an amus, a great, a lub, a cabid, an em, a natural, a tip, a carros, a pass, a baromet, a minimus, a ari, a fall, a bullet, a knot, a pot, a rep, a curat, a mat, a parten, a pelt, a gatewary, a less, a pay, a cap, a rag, a fat, a hall, a gamb, a dah, a cam, a tube, a day, a butt, a waterfall, a palinta, a mat, a floor, a kios, a van, a move, a nih, a drew, a proper, a call, a war, a sty, a gum, a yap, a cam, a ray, an as, a taj, a wax, a pyw, a cat, a valley, a dreb, a lion, a saga, a flat, a caitump, a pooh, a rail, a column, a dairyman, a hitzer, a canal ---Panama!

Elaine has a 2^n by 2^n kitchen. One square of it is covered by a 1 by 1 refrigerator. Tile the kitchen with these kinds of tiles:

Creating a recursive method
public static String deblank(String s) {
    if (s.length == 0)
        return s;
    // {s is not empty}
    if (s.charAt(0) is a blank)
        return deblank(s.substring(1));
    // {s is not empty and s.charAt(0) is not a blank}
    return s.charAt(0) +
        deblank(s.substring(1));
}

Check the four points:
0. Precise specification?
1. Base case: correct?
2. Recursive case: progress toward termination?
3. Recursive case: correct?

Check palindrome-hood
A String with at least two characters is a palindrome if
(0) its first and last characters are equal, and
(1) chars between first & last form a palindrome:
e.g. A MAN A PLAN A CANAL PANAMA
/** = "s is a palindrome" */
public boolean isPal(String s) {
    if (s.length <= 1)
        return true;
    // { s has at least two characters }
    return s.charAt(0) == s.charAt(s.length()-1)  &&
        isPal(s.substring(1, s.length()-1));
}

Tiling Elaine's kitchen
Elaine has a 2^n by 2^n kitchen. One square of it is covered by a 1 by 1 refrigerator. Tile the kitchen with these kinds of tiles: