

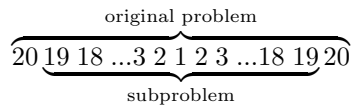
CS163 Homework: Recursion

Due: October 8

Recall the following guidelines when writing a recursive solution to a problem.

- Describe how a simpler and similar subproblem is defined in the context of the original problem. The subproblem is to be solved using a recursive call(s).
- Determine the parameter(s) that distinguish the subproblem from the original problem
- Describe the simplest instance of the problem
- Specify the pre- and post-condition of the method (the "contract")
- Write the recursive method that incorporates the above elements. When additional variables (besides the parameters) are needed, your solution must use only **local** variables. **Avoid using global variables in recursive programming!**

Example: Print descending and ascending sequence of integers:



- The subproblem is defined by excluding the first and the last number from the original sequence
- The parameter that distinguishes the subproblem from the original problem is the first (and last) number in the sequence
- The simplest instance is when the sequence is a singleton 1

```
1  /**
2   * pre-cond : N is positive
3   * post-cond: a descending sequence from N to 1 followed by
4   *           an ascending sequence from 1 to N is printed
5   */
6  public void printDownUp (int N)
7  {
8     if (N == 1) /* solve the simplest problem */
9         System.out.print (N);
10    else {
11        System.out.print (N + " ");
12        printDownUp (N - 1);
13        System.out.print (" " + N);
14    }
15 }
```

1. A pizza is usually cut into $2N$ uniform slices by rolling a pizza cutter N times through the center of the pizza. However, if the cutting pattern is sloppy (the cutter does not go through a common intersection point), we will get more (irregular) slices. For instance, with 3 cuts we can get 7 irregular slices and with 5 cuts we can get 16 irregular slices. Write a recursive formula (no Java code needed!) for $S(k)$ to determine the number of pizza slices (given the number of cuts k).

What is the geometric interpretation of the recursive formula! Hint: *Think of one cut as one infinite line on a plane. When k cuts have been made, observe how the plane changes when the $k + 1$ -th is being made.*

2. Write a method, `void plusMinus (int N, boolean isPlus)` that prints a descending and ascending sequence with alternating plus and minus signs.

For instance `plusMinus(6, true)` will print:
`+ 6 - 5 + 4 - 3 + 2 - 1 + 1 - 2 + 3 - 4 + 5 - 6`
and `plusMinus(6, false)` will print:
`- 6 + 5 - 4 + 3 - 2 + 1 - 1 + 2 - 3 + 4 - 5 + 6`

3. Write a recursive method to find the smallest value in an array of double-precision floating point numbers. Use **two recursive calls**, one call operates on the lower half of the array and the other on the upper half. The method returns **the index** of the smallest value, not the smallest value. If more than one such element are found in the array, return the lowest index.

```
public int findSmallest (double[] x, int L, int R) { }
```

the two parameters L and R specify the indices of the leftmost and rightmost elements (inclusive). So, for an array `myData` of N elements, the initial method invocation is

```
int idx = findSmallest (myData, 0, myData.length - 1);  
System.out.printf ("The smallest element is at index %d\n", idx);
```

For instance, `findSmallest ({123.12, 105.04, -30.1, -10.0, 84.4, -13.12, 94.4}, 0, 6)` returns 2. Avoid creating a copy of the array to solve the problem. Notice that the method has an integer return value. Your recursive solution should use this return value properly.

4. Write a recursive method that returns the currency format of the given integer value (*val*). All intermediate operations must be carried out as integer arithmetic. The only time you are allowed to use string operations is to return the result. You may use the `String.valueOf()` or `Long.toString()` methods.

```
public String formatAsCurrency(long val) { }
```

For this question, make sure your method works correctly for numbers that are multiple of 1000 or has consecutive zeros. Examples:

```
formatAsCurrency(5481000L) returns "5,481,000"  
formatAsCurrency(32000457L) returns "32,000,457"  
formatAsCurrency(67321L) returns "67,321"  
formatAsCurrency(80005L) returns "80,005"  
formatAsCurrency(434L) returns "434"
```

5. Choose one question from the following JavaBat Recur1 problem set: `count8`, `changePi`, `endX`, `countPairs`

6. Choose one question from the following JavaBat Recur1 problem set: `stringClean`, `countHi2`, `strCopies`, `strDist`
7. (Extra credit) For each problem you choose from JavaBat Recur2 problem set, you will earn 10 points.