

## Recursion & Methods

### **What To Do:**

Follow each step carefully. As you complete the lab, submit the source files (.java) problems to the autograder (link is in the Canvas portal). After finishing, please submit your work to the autograder and let one of the TAs know.

**For this lab, please place each method inside its own class file labeled as ProblemX, where X is the problem number. The accompanying test files should be named ProblemXTest.**

### **Problem 1:**

- (i) Design the recursive `String replaceAB(String s)` method that replaces any occurrence of the character 'A' with the character 'B' in a given string `s`.
- (ii) Design the tail recursive `String replaceABTR(String s)` method that solves the same problem as `replaceAB`, but instead uses tail recursion. Remember to include the relevant access modifiers!

**Problem 2:**

Design the `isNestedParenthesesTR` tail recursive method, which receives a string and determines if its parentheses pairs are “balanced.” A pair of parentheses is balanced if it is a nesting of zero or more pairs of parenthesis, like “`()`” or “`((()))`.” Note that pairs like “`((()))`” will not be tested.

**Problem 3:**

A *factorion* is a number that is equal to the sum of the factorials of its digits. For example, 145 is a factorion because  $1! + 4! + 5! = 1 + 24 + 120 = 145$ .

- (a) Design the standard recursive `isFactorion` method, which receives an integer  $n$  and returns whether or not it is a factorion.
- (b) Design the `isFactorionTR` tail recursive method, which solves the same problem as part (a), but uses tail recursion.