Here are the problems we will go over in Week 2 of the bootcamp. They are based on the material we covered in Week 1, namely, primitive types, control constructs, and reference types.

Each problem entails writing a function. The signature specifies the name of the class. Each specified function is to be public and static. Problems marked with a $\star$ are more challenging.

You do not need to check if the input is valid, e.g., for the interconverting strings and integers problem, you can assume the string consists of digits, and, optionally, a $-$ sign starting out.

Problem 1. W-1.19

Task: Write a function `M1.count(int n)` that returns the number of pairs of positive integers (a, b) such that $a < b < 1000$ and $(a^2 + b^2 + 1)/(ab)$ is an integer. (You may ignore overflow checking.)

Problem 2. W-1.20 $\star$

Task: In the following puzzle, each of the letters is assigned a digit. Write a function that finds a solution, if one exists, as illustrated below. Your solution should consist of an array of integers of length 26—entry 0 is the mapping from 'a' to a digit, entry 1 is the mapping from 'b' to a digit, etc. Signature: `int[] M2.code(String a, String b, String c)`, where a, b, c are strings of digits, and we want $a + b = c$.

```
   MARK     A=1 W=2 N=3 E=5      9147
+ ALLEN     L=6 K=7 M=9 S=0   + 16653
  -----                         -----
  WEISS                         25800
```

Problem 3. W-2.11

Task: Write a function `boolean S1.isPrefix(String s1, String s2)` that returns true if and only if $s1$ is a prefix is $s2$. (The string p is a prefix of string x if and only iff $x = p + s$ for some string s—here $+$ is string concatenation. For example abc is a prefix of $abcdef$, but abd is not a prefix of $abcdef$. You cannot use any String library functions other than `charAt()`.

Problem 4. W-2.23

Task: Write a function `boolean M3.checkMonotone(int[][] A)` which returns true if and only if the entries in each row of A appear in increasing order, and the entries in each column of A appear in increasing order. (Assume all rows of A have the same length.)

For example, your function should return true for $A1 = \{\{1,2,3\},\{4,5,6\},\{7,8,9\}\}$ and $A2 = \{\{0,0\},\{0,1\}\}$ and false for $A3 = \{\{1,2,3\},\{4,0,6\},\{7,8,9\}\}$.

Problem 5. W-2.31

Task: Write a function `int S2.wc(String s)` that returns the number of words in s. A word w is a nonempty substring of s that is made up of nonblank characters; to w's left should be a blank or it should start s, and to its right should be a blank, or it should end s. For example, you should return 3 for `"foo bar widget"`, 1 for `"california"`, and 2 for `"  a b,c   d:e "`.

Problem 6. W-2.35

Task: Write a function `String S3.fileToString(String filename)` that takes the name of a file and returns its contents as a string. Then write a function `String S3.computeGrades(String scores)` which calculates the grades for a class. Your function should take its input as a string which is a sequence of lines formatted as follows:

```
LastName:FirstName:Exam1:Exam2:Exam3
```

The exams are scored on 100 and each entry is an integer score. The weights are 25% for the first exam, 30% for the second exam, and 45% for the third exam. Your function should return a string in the following format, one line per student.

```
LastName FirstName LetterGrade
```

A 90% or higher is an A; after that, 80% or higher is B, etc.

For example, if the `scores` input is

```
Doe:John:100:100:100
Pantz:Smartee:80:90:80
```

the returned value should be

```
Doe John A
Pantz Smartee B
```

Be careful with newlines. In UNIX and MacOS, you can use the string `"\n"` to encode a newline, but on Windows you need to use `"\r\n"`. See this StackOverflow article for a platform independent way of handling newlines:

`http://stackoverflow.com/questions/5649362/java-n-delimiters-issue`

Problem 7. Interconverting Strings and ints

A string is a sequence of characters. A string may encode an integer, e.g., “123” encodes 123. In this problem, you are to implement functions that take a string representing an integer and return the corresponding integer, and vice versa.

Your code should handle negative integers. It should throw an exception if the string does not encode an integer, e.g., “123abc” is not a valid encoding.

Languages such as C++ and Java have library functions for performing this conversion—`stoi` in C++ and `parseInt` in Java go from strings to integers; `to_string` in C++ and `toString` in Java go from integers to strings. You cannot use these functions. (Imagine you are implementing the corresponding library.)

Task: Implement string/integer inter-conversion functions. Use the following function signatures: `String intToString(int x)` and `int stringToInt(String s)`.

Problem 8. Checking if rectangles intersect

Call a rectangle R whose sides are parallel to the x-axis and y-axis *xy-aligned*. Such a rectangle is characterized by its left-most lower point (R_x, R_y), its width R_w and its height R_h.

Task: Let R and S be xy-aligned rectangles in the Cartesian plane. Write a function which tests if R and S have a nonempty intersection. If the intersection is nonempty, return the rectangle formed by their intersection, otherwise return `null`. Use the following signature: `Rectangle intersect(Rectangle a, Rectangle b)`. The rectangle class is as follows:

```
class Rectangle {
  // fields
  int lowerLeftX;
  int lowerLeftY;
  int Xlength;
  int Ylength;
  // add any code you want
}
```

Problem 9. Enumerating primes ⋆

A natural number is called a prime if it is bigger than 1 and has no divisors other than 1 and itself.

Task: Write a function that takes a single positive integer argument n ($n \geq 2$) and returns all the primes between 1 and n. Use the following signature: `int[] primes(int n)`.

Problem 10. Invert a permutation ⋆

Every one-to-one onto mapping is invertible, i.e., if f is one-to-one onto, then there exists a unique function f^{-1} such that $f^{-1}(f(x)) = x$. In particular, for any permutation Π, there exists a unique permutation Π^{-1} that is the inverse of Π.

Given a permutation represented by an array A, you can compute its inverse B by simply assigning $B[A[i]] = i$ for all values of i.

Task: Given an array A of integers representing a permutation Π, update A to represent Π^{-1} using only constant additional storage. Use the following function signature: `void invert(int[] A)`.

Problem 11. Run-length encoding

Run-length encoding (RLE) compression offers a fast way to do efficient on-the-flight compression and decompression of strings. The idea is simple—encode successive repeated characters by the repetition count and the character. For example, the RLE of "aaaabcccaa" is "4a1b3c2a". The decoding of "3e4f2e" returns "eeeffffee".

Task: Implement run-length encoding and decoding functions. Assume the string to be encoded consists of letters of the alphabet, with no digits, and the string to be decoded is a valid encoding. Use the following signatures: `String encode(String u)`, `String decode(String e)`.

Problem 12. Replace and remove

Consider the following two rules that are to be applied to strings over the alphabets {"a", "b", "c", "d"}.

(a) Replace each "a" by "dd".

(b) Delete each "b".

It is straightforward to implement a function that takes a char array s as an argument, and applies these rules to s if the function can allocate a new char array whose length equals that of s.

Task: Write a function which takes as input a char array s, and removes each "b" and replaces each "a" by "dd". You cannot allocate additional storage, i.e., no calls to `new`. Assume s is stored in an array that has enough space for the final result. Use the following signature `void removeAndReplace(char[] s)`.

Problem 13. BigInteger multiplication ⋆

Certain applications require arbitrary precision arithmetic. One way to achieve this is to use strings to represent integers, e.g., with one digit or negative sign per character entry, with the most significant digit appearing first.

Task: Write a function that takes two strings representing integers, and returns an integer representing their product. Use the following signature: `String bigMultiply(String X, String Y)`.

Problem 14. Dutch National Flag partitioning ⋆

Write a function that takes an integer array A and an index i into A, and rearranges the elements such that all elements less than $A[i]$ appear first, followed by elements equal to $A[i]$, followed by elements greater than $A[i]$. Your algorithm should run in time proportional to the length of A and should not allocate additional memory. Use the following signature: `void DNF(int[] A, int i)`.

Problem 15. Sudoku checker

Sudoku is a popular logic-based combinatorial number placement puzzle. The objective is to fill a 9×9 grid with digits subject to the constraint that each column, each row, and each of the nine 3×3 sub-grids that compose the grid contains unique integers in $[1, 9]$.

Task: Check whether a 9×9 2D array representing a partially completed Sudoku is valid. Specifically, check that no row, column, and 3×3 2D subarray contains duplicates. A 0-value in the 2D array indicates that entry is blank; every other entry is in $[1, 9]$. Use the following signature: `boolean sudokuChecker(int[][] A)`.

Problem 16. Parity checker ⋆

The parity of a sequence of bits is 1 if the number of 1s in the sequence is odd; otherwise, it is 0. Parity checks are used to detect single bit errors in data storage and communication. It is fairly straightforward to write code that computes the parity of a single 64-bit nonnegative integer.

Task: Write a function for computing the parity of a long. Assume the function will be called frequently, with many different arguments. Use the following signature: `int parity(long x)`.