## Specification

Submit a `.java` file named `YourNamePset2.java`. Your file should **NOT** execute any code. You are just filling them in. When I run your program there should be no errors and no output. You may choose to test your methods as you go inside of a `main` method, but be sure to delete the main method before submitting. Comment your problems with the problem number above the method header. Do not use built-in methods that are not in the Java Quick Reference guide.

## Problem 1

Given two strings, append them together (known as "concatenation") and return the result. However, if the strings are different lengths, omit characters from the longer string so it is the same length as the shorter string. Remove characters from the start of the longer string. The strings may be any length.

```
minCat("Hello", "Hi")   // "loHi"
minCat("Hello", "java") // "ellojava"
minCat("java", "Hello") // "javaello"
```

## Problem 2

Return the number of times that the string "java" appears anywhere in the given string, except it should accept any letter for the 'v', so "jala" and "jara" count.

```
countJava("aaajavabbb") // 1
countJava("jalaxxjara") // 2
```

## Problem 3

A palindrome is a word spelled the same way forwards and backwards. Given a string `str`, return `true` if `str` is a palindrome, and `false` otherwise.

```
isPalindrome("racecar") // true
isPalindrome("mom")     // true
isPalindrome("number")  // false
```

## Problem 4

Given a string `str` that includes numbers, add all the numbers in a given string together and return the sum. Note that it is numbers, and not *digits*. For example `"abc123"` would be `123`, not `6`.

```
sumString("abc1n3n4") // 8
sumString("abc13nm4") // 17
sumString("aba123")   // 123
```

## Problem 5

Write a method that returns `true` if for every `*` (star) in the string, the letters on the left and right are the same. Stars in the front or end of the string do not contribute.

```
sameStarChars("xy*zzy")   // false
sameStarChars("ox*xa*az") // true
sameStarChars("*oxox*x*") // true
```

## Problem 6

Two words are anagrams if they use exactly the same letters but rearranged. Write a method to check if two strings `str1` and `str2` are anagrams of each other.

```
areAnagrams("listen", "silent")     // true
areAnagrams("triangle", "integral") // true
areAnagrams("hello", "world")       // false
```

## Problem 7

Write a method to find the length of the longest substring of a string `str` without repeating characters.

```
longSubStringLen("abcabcbb") // 3 (abc)
longSubStringLen("bbbbbb")   // 1 (b)
longSubStringLen("pwwkew")   // 3 (kew)
```

## Problem 8

A string can be rotated by moving letters from the front to the back. The string `"abc"` can be rotated to `"bca"` or `"cab"`. Write a method to check if one string `str1` is a rotation of another `str2`.

```
areRotations("abcd", "cdab")   // true
areRotations("hello", "llohe") // true
areRotations("water", "atwar") // false
```

## Problem 9

Write a method that returns the longest word in a sentence. In the event of a tie, return the last one.

```
longestWord("the quick green fox") // green
longestWord("ap csa is easy")      // easy
longestWord("strings are tricky")  // strings
```

## Problem 10

Harshad numbers are those that are divisible by the sum of their digits. Write a method that returns the number of Harshad numbers between `1` and `n` (inclusive).

```
isHarshad(1000)    // 213
isHarshad(1000000) // 95428
```

## Problem 11

In the first assignment you used the built in square root function provided by the programming language. Not only did Newton discover a majority of classical mechanics, and arguably calculus, he also discovered a way to find the square root of a number. `Newton's Method` as it's called works for positive values of $x$ in the following way. Start with an estimate $t$ equal to $x$. If $|t^2 - x|$ is less than some threshold, $t$ is the square root of $x$. If it is not within the threshold, replace $t$ with the average of $t$ and $\frac{x}{t}$. Write a method that calculates the square root of a number up to $\pm 10^{-15}$ of it's value.

```
sqrt(12039) // 109.72...
sqrt(25)    // 5.0
sqrt(0.4)   // .63245...
```

## Problem 12

A finite sum in mathematics is one that adds a finite number of terms. These sums can have many many terms and can take a long time to do by hand. The Harmonic Sum $H_n$ is given by $H_n = \sum_{k=1}^{n} k^{-1} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + ... + \frac{1}{n}$. Write a method that returns the finite sum of the first 'n' harmonic numbers.

```
harmonicSum(10)    // 2.9289
harmonicSum(1000)  // 7.48547
harmonicSum(10000) // 9.78760
```

## Problem 13

The Harmonic sum that you just looked at is said to be `divergent`. This is because as we add more and more terms, the sum becomes larger and larger. In 1650, Pietro Mengoli presented a problem called the Basel problem in which he stated that the infinite sum $\sum_{k=1}^{k=\infty} k^{-2}$ had a finite value. Euler solved the problem exactly and demonstrated that $\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + ... + \frac{1}{n^2} = \frac{\pi^2}{6}$ when $n$ goes to $\infty$. Use this fact to write a method that returns an estimate for $\pi$. Be careful to not go too high with $n$ or else you'll get an overflow error. Java does not handle large numbers very well. With a value of 10000 I got an output of 3.141497 which is close enough for me. If you are interested in getting a value that is closer to $\pi$, you may use the `BigDecimal` object from the `java.Math` class but this is not necessary. Taking the square root of a `BigDecimal` object requires doing Newton's method for this particular data type.

```
estimatePi() // something close to pi
```

## Problem 14

Write a function that takes an integer $n$ and displays a filled and hollow square placed next to each other of length $n$ and width $n$.

```
makeSquares(5)
***** *****
***** *   *
***** *   *
***** *   *
***** *****
```

## Problem 15

Write a function that takes an integer $n$ and displays a pyramid of $n$ levels.

```
makePyramid(4)
   *
  ***
 *****
*******
```

## Problem 16

Write a function that takes an integer $n$ and displays a hollow diamond with side length $n$.

```
makeDiamond(3)
  *
 * *
*   *
 * *
  *
```

## Problem 17

One of the goals of the Pokemon game is to "catch them all." In the game, you walk around and "encounter" a particular pokemon that you can catch. In this question, we are interested in estimating just how many encounters you'd have to have to catch them all. The game itself is a bit complicated, so we'll make some assumptions. Let's assume there are 150 pokemon and they all show up with equal probability. Next, we assume that upon encountering a pokemon there is a 100% success rate in catching that particular pokemon. Use a Monte Carlo simulaton to determine the total number of encounters you'd expect to have to catch all 150 pokemon under these rules.

```
nEncounters() // some integer number!
```

## Problem 18

Happy birthday! It seems like a rare occurance that you meet someone with your exact birthday, but it probably will happen at some point in your life! In this problem, we're interested in estimating just how many people have to be in the same room in order to have at least two that share a birthday. In this problem we'll make some assumptions. We'll assume that being born on any day of the year is equally likely, and that there are no leap years. Use a Monte Carlo simulation to determine the minimum expected number of people that have to be in the same room to have two that share a birthday.

```
sameBirthday() // some integer number!
```

## Problem 19

There is a certain disease that affects x% of the population, and a medical test produces a correct result g% of the time. Use a Monte Carlo simulation to determine the probability that a person is actually sick if they test positive.

```
// @params (x, g)
pHasDisease(.01, .97) // some double number!
```

## Problem 20

A ball is thrown from a building of height $h$ with a velocity $vi$ at an angle $\theta$ to the horizontal. Assume there is gravity $g$ on the planet. Using the equations $y = y_0 + v_{0y} * t + \frac{1}{2}gt^2$ and $x = v_{0x} * t$ print out the x and y coordinates every $dt$ seconds in a format that you could save to a file and plot to see the trajectory. You may stop the calculation when the ball hits the ground. Return the range of the motion. You may estimate the range as the distance in the x direction the ball travels the time step after, or before, the ball hits the ground.

```
// @params (h, vi, g, theta, dt)
getTraj(20, 5, -9.8, 65, 0.1) // 5.494037
Output:
0.0 20.0
0.21130913087034975 20.404153893518323
0.4226182617406995 20.71030778703665
0.6339273926110492 20.918461680554977
0.845236523481399 21.0286155740733
1.0565456543517486 21.040769467591623
1.2678547852220983 20.954923361109948
...
5.071419140888395 2.651693444439779
5.282728271758745 0.7038473379581021
5.494037402629095 -1.3419987685235704
```