## Learning Outcomes:

- Deepen your understanding of recursive functions
- Collect and plot data to visualize the time complexity of different algorithms
- Formulate a recursive algorithm
- Demonstrate that the functionality provided by recursive algorithms can also be provided in an iterative way

## Challenge

Create a single file called **`Recursion.java`** that includes all the methods described below.

1. Write two public static methods **`fib1`**and **`fib2`** as described in the specifications below.
   Collect, plot, and analyze the running times of **`fib1`** and **`fib2`**.
2. Write two public static methods **`palindrome1`** and **`palindrome2`** as described in the specifications below.

## Specification

### 1. Fibonacci Numbers:

When Fibonacci introduced the number sequence, he started with 1, 1, 2, 3, 5 … However, in modern math the sequence is often listed as 0, 1, 1, 2, 3… For this assignment we use **0** as the starting point of the Fibonacci sequence.

**Write two recursive methods `fib1`and `fib2`.**

Both `fib1` and `fib2` have an int parameter `n` and return a value of type `long` that is the $n^{th}$ Fibonacci number

Passing a value less than 1 doesn't make sense (there is no $-2^{nd}$ or $0^{th}$ Fibonacci number)

In those cases the methods should throw an **`IllegalArgumentException`.**

```
fib1(0)  .. IllegalArgumentException,    fib1(-2) .. IllegalArgumentException
fib1(1)  .. 0,    fib1(2) .. 1,    fib1(5) .. 3,
```

- **`fib1`** is a simple, straight-forward recursive implementation, that doesn't store any values (it keeps re-calculating the same values over and over again)
- **`fib2`** is an improved recursive version, that stores and reuses values that have already been calculated.
  During class we watched part of a video that explained recursion with memorization. To demonstrate the idea it included a C++ implementation that used an array outside of the method.
  I want you to write a Java implementation that declares the array inside a 'wrapper' method.

```
long fib2(int n) {   // fib2 is the name of a 'wrapper' method
    ...
    // declare array ⌂
    f(n, ⌂);        //  the recursive method call; the second argument is the array
}
```

When the 2 methods are implemented and tested (I recommend to use unit tests for this purpose) run the methods a number of times and measuer the elapsed time.
Create 2 tables - onefor `fib1` and one `fib2`. Both tables should list the values of n and the corresponding elapsed times.
Based on 5 – 7 entries from each of the tables draw the graphs for `fib1` and `fib2`. (tables and graphs should be clearly labeled)

2. **Palindrome:**

Any loop can be replaced by a recursive method **and** any recursive method can be replaced by an iterative counterpart.

**Write two recursive methods `palindrome1` and `palindrome2.`**

Both `palindrome1` and `palindrome2` have a single argument of type `String` and return true or false, depending on whether the string passed was a palindrome or not.

- **`palindrome1`** is a 'wrapper' method that calls a recursive implementation of palindrome (see above)
- **`palindrome2`**is an iterative implementation of palindrom (uses a loop and doesn't call itself)

A palindrome is the same when read forward and backward. E.g. *kayak* and *Dad* are palindroms. *Aunt* is not
Befure you turn in your methods make sure they are tested. You could write a test client with a main method but I recommend to use unit tests. (When I grade the assignment I will run it against a number of unit tests)

## Turning in:

Turn in 2 files

1. A jar or zip file that includes one single Java file called **`Recursion.java`**
2. A file that includes the 2 tables and 2 graphs. (pdf, docx, jpg) . . . separate from the zip file

**Please Check:**
Recursion.java should include all the methods described above.
Make sure the fib1, fib2, palindrome1, and palindrome2 are declare public and static.
Double check the specified method signatures – name, parameters, return type.
Why does it matter? That is important so that different programs can work together (in this case that your code will work with my unit tests.)