# COMP 2140 Lab 1 — Recursion Versus Iteration

Helen Cameron

Week of September 19, 2016

## Objective

To compare the efficiency of a recursive implementation of an algorithm and an iterative version.

## The Cado Numbers

The Cado numbers are defined as follows:

$$\begin{aligned} C(0) &= 1 \\ C(1) &= 1 \\ C(2) &= 1 \\ C(n) &= C(C(n-1)) + C(n - C(n-1)), \text{ for } n \geq 3. \end{aligned}$$

For example, here is how you would compute $C(3)$:

$$\begin{aligned} C(3) &= C(C(3-1)) + C(3 - C(3-1)) \\ &= C(C(2)) + C(3 - C(2)) \\ &= C(1) + C(3-1) \quad \text{because } C(2) = 1 \\ &= 1 + C(2) \quad \text{because } C(1) = 1 \\ &= 1 + 1 \quad \text{because } C(2) = 1 \\ &= 2 \end{aligned}$$

Here are some values in the series:

| $n$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $C(n)$ | 1 | 1 | 1 | 2 | 2 | 3 | 4 | 4 | 4 | 5 | 6 | 7 |

## Exercise

Write a recursive method to compute the $n^{\text{th}}$ Cado number, for $n \geq 0$.

Then write an iterative method that uses a loop (instead of recursion) and an array (to store Cado numbers as you compute them) to compute the $n^{\text{th}}$ Cado number, for $n \geq 0$. The idea: put $C(i)$ into position $i$ of the array, from 0 to $n$.

Finally, write a `main` method that calls each of the above methods to find $C(100)$. The `main` method should report how much time each of the two methods takes to compute $C(100)$; the following example code shows you how your `main` method might do the timing:

```
  long startTime, endTime, elapsedTime;
...
  startTime = System.nanoTime();
  // call one Cado method here
  endTime = System.nanoTime();
  elapsedTime = endTime - startTime;
```

Note that `System.nanoTime()` gives nanosecond precision, but not necessarily nanosecond accuracy. To be able to use `System.nanoTime()` in your code, you must also include the following line at the beginning of your class:

```
import java.util.*;
```