Exercises Week 1

Exercise 1: `Cores.java`. Create as many threads as there are cores.

Use a for loop to create as many threads as there are cores in your computer. Give each of the threads their unique ID's.

Hint: To get the number of available cores in your computer, use this oneliner:

```
int cores = Runtime.getRuntime().availableProcessors();
```

Exercise 2: Calculate the speedup.

You have written two algorithms for finding the largest number in an array; one parallel algorithm and one sequential algorithm. Using an array with 100 million elements, you time each of them and find out that the sequential algorithm takes 10 seconds to complete and the parallel takes 5 seconds to complete. You have a computer with 4 cores.

1. What is the speedup of the parallel algorithm?
2. What is the largest speedup you can obtain on your computer?
3. What does it say about the parallel algorithm if the speedup is below 1?

Exercise 3: `Sleep.java`

Create a small program that runs three threads. Make the first thread sleep for one second, the second thread sleep for two seconds, and the third thread sleep for three seconds. Write suitable output to terminal that shows when each thread sleeps and wakes up.

Exercise 4: Calculate the runtime.

Extend Exercise 3. Time the program that you created and print the runtime (in **seconds**) to the terminal. How much time did it use?

Hint: To time a specific portion of your program you can do:

```
long t0 = System.nanoTime(); // gets time in nanoseconds
// code to be timed
long t1 = System.nanoTime();
long runtime = t1 - t0 // runtime in nanoseconds
```

If the time is not below 6 seconds, exercise 3 is wrong (unless you only have 1 core). Try to change your solution to exercise 3 so that the runtime is below 6 seconds. If the time is below 3 seconds, something is wrong. Hint: think about which thread is calculating the time. Hint: use `join()`.