Exercises Week 2

Exercise 0: Get to know your classmates' computers!

Turn on your camera, say hello to your classmates and tell them about your computer! Tell them when you bought it, what brand it is, what operating system(s) you are running (maybe you have dual boot?) and whether it is your baby. Don't forget to tell them how many cores you have as well!

Exercises 1 and 2 are created by Eric Jul "*etter opplegg av Arne Maus*" (translated to English by Shiela).

Exercise 1: `FindMax.java`

In this exercise we are going to find the largest element in an integer array *a[]* of size *n*. The array's elements should be made up of randomly generated integers. Use Java's `Random` class from the package `java.util` and the non-static method `nextInt(n)` to place randomly generated integers in *a[]*.

We are going to test all our algorithms on arrays of sizes *n* = 100, 1000, …, 10 million.

a) Sequential algorithm

First make a sequential solution, time this with `System.nanoTime()` and print out the runtime in **milliseconds**. To get the milliseconds in *double* and not *int*, divide the nanoseconds by 1000000.0. Make a table of the results.

b) Parallel algorithm

We are going to look at two ways to parallelize our problem. Let's say we have *k* threads:

1. Divide the array so that `Thread 0` gets the first *n/k* elements, `Thread 1` the next *n/k* elements, and so on. The last thread, `Thread k-1`, should get the rest of the array.

2. Divide the array so that `Thread 0` gets elements *0, k, 2k, 3k, …*; `Thread 1` gets elements *1, k+1, 2k+1, …*; and so on. Remember to check that you do not go beyond the end of the array.

For each of these method, the threads need to update a shared variable called `globalMax`. Here you can test two strategies:

1. Use a `synchronized` method that all the threads call for each element they are responsible for. This method should update `globalMax` each time it is called.

> A `synchronized` method is thread-safe as it can only be invoked by one thread at a time and ensures that changes to the state of an object is seen by all threads.

2. Let all the threads have their own variable `localMax` which they use to find the largest element in their part of the array. Only after a thread has found its local max, does it use the `synchronized` method to update the `globalMax`.

Make a small report about the runtimes of the 4 parallel variants and the sequential algorithm. Calculate the speedup and comment on whether it seems possible to obtain a speedup > 1 for any of the 4 alternatives.

Discuss with your classmates why some of the variants are especially slow or especially fast.

Hint: To create a `synchronized` method, simply use the `synchronized` keyword:

```
synchronized void updateGlobalMax(int num) {
  ...
}
```

Exercise 2: `FindSum.java`

If you have time, do the same as in Exercise 1 for an algorithm that finds **the sum of all the elements in an array**. Make a small report with the runtimes. Do you arrive at the same conclusions as you did with the `FindMax` problem?