EXAM EXAMPLE

This exam example for PG4200 is composed of 10 questions/exercises. Each question is worth 10 points, for a total of 100 points. You have 3 hours to answer as many of them as possible.

All the questions are written in English. To answer these questions, it is preferred that you do it in English. However, any other language officially recognized by Kristiania (e.g., Norwegian) is obviously acceptable.

Each question/exercise might actually be composed of more than 1 question/part (e.g., several sentences ending with a "?"). You need to address all of them.

When you see required to explain "*in details*", just a couple of sentences will likely not be enough for answering in details. You might expect to need to write at least half-page, but likely no more than a full one. However, these are just high level indications, and of course the amount needed to answer in details depends on the question.

When writing code on a piece of paper or in a text editor (not an IDE), it is obviously expected that there will be syntactic errors. Those will not reduce your grade. Still, the more you can be close to the actual Java syntax, the better. You are **NOT** allowed to use pseudo-code or different programming languages. If you do not remember the exact name for a specific class/method, use a meaningful name that somehow reflects the needed functionality.

When you are asked to implement a class extending a given interface, you will also need to implement any *private/protected* method required to be called from the specified *public* methods.

As discussed in class, you are **NOT** allowed to use "*delegate*" implementations (unless explicitly requested). For example, if you need to implement a *Map*, you cannot rely on an existing implementation, and then call methods on it, e.g. writing something like "*V get(K k){return delegate.get(k);}*" is not allowed.

1) Consider 4 functions representing runtimes of 4 different algorithms for the same problem, for which we have the following bounds based on the input size *n*. What can you say about their relative performance? Which one is best? Which one would you choose? Motivate your answer in *details*.

$f(n)$ = O($log\ n$), $g(n)$ = Ω(n), $t(n)$ = Ω($n\ log\ n$), $k(n)$ = O(n^2)

2) Explain the main differences between the *List* and the *Set* data structures.

3) What are the main properties that differentiate *Red-Black Trees* from *Binary Search Trees*?

4) Explain what is an *immutable* object. Why the *keys* in a Map (and values in a Set) data structure must be immutable? Explain in *details*, in particular what could happen if such keys are modified after an insertion.

5) In relation to streams in data structures, explain in *details* what are the main differences between the *map* and *flatMap* methods and what they do.

6) If you need to sort some data, which of these two algorithms would you use: *Bubble Sort* or *Merge Sort*? Are they equivalent? Or is one always better than the other? Or does it depend on the context? In this latter case, in which contexts would one algorithm be better than the other, and vice-versa? Explain *in details*.

7) In the context of Genetic Algorithms, explain in *details* what *Elitism* is, what is used for, how it works, and why it is important.

8) Given the following method signature, implement a Quick Sort algorithm.

```
public <T extends Comparable<T>> void sort(T[] array)
```

9) Given the following interface, implement a concrete Hash Map for it:

```
public interface MyHashMap<K, V> {

    void put(K key, V value);

    void delete(K key);

    V get(K key);
}
```

10) Considering the following beginning of the class for

```
public class UndirectedGraph<V> implements  Graph<V>{
   /**
    * Key -> a vertex in the graph
    * Value -> set of all vertices that connect to the Key,
    *          ie the Key is the "from"/"source"
    */
   protected Map<V, Set<V>> graph = new HashMap<>();
```

Implement a method for a Breadth-First Search with signature

```
public List<V> findPathBFS(V start, V end)
```

Assume that the rest of that class is implemented (as seen in class), but that the method *findPathBFS* is missing and you need to implement it.

THIS MARKS THE END OF THE EXAM TEXT