Exercise 1 (Range Queries, 3+3 Credits)

kd-trees are perfectly suited for range queries.

a) Describe an algorithm that delivers all points in a pre-defined axis-aligned query rectangle.

b) What is the (output-sensitive) running time of this algorithm?

Exercise 2 (kd-Trees and Triangles, 3+4 Credits)

It is also possible to use kd-trees for triangular range queries.

a) Show that the running time for a triangular range query in a 2-dimensional kd-tree can be linear in the worst case, even if the triangle does not contain a single point of the point set.

Hint: Look at points on the line $y = x$.

b) In the following we will only consider triangles in the plane with edges that are either axis-aligned (horizontally or vertically) or that have a gradient of −1 or +1.
Define a data structure that can answer range-queries for this particular triangles more efficiently. 

Hint: Extend the 2-dimensional kd-tree.

Exercise 3 (Beyond General Position, 2+3+2 Credits)

Until now, we mainly considered a good distribution of the points: they are in the so called general position, i.e. all points differ in their x- and y-coordinates. Consequently, it is always possible to split them with an axis-aligned splitting line into two subsets of (almost) the same size. Unfortunately, in real-world data sets, points hardly have this nice property. Therefore, we also have to think about the implications that such bad data sets have on the performance of our data structures.

a) Describe an example for a set of points in 2D that you cannot divide into two sub-sets of the same size with any axis-aligned line (Obviously, your set should have even size).

b) Let $R$ be a set of points in the 2D plane. Show that there always exist an axis-aligned line that does not contain any point of $R$ but splits $R$ into two subsets where the smaller subset contains at least $\frac{n-1}{2}$ points. Is this bound tight?

c) Has the bound for the split in part (b) any negative implications on the theoretical running time of range queries in kd-trees?