Question 1: Basics. (20/80 marks)

Question 1.a. (10 marks)

Provide a definition of a function called `translate` that translates a given 3d point in a given direction. The first argument of the function is the point and the second argument is the direction. The x, y, and z coordinates of both parameters are arbitrary precision integers. You may carry out the translation by pairwise adding the x, the y, and the z coordinates. E.g. `translate (1,2,0) (3,7,0)` results in `(4,9,0)`.

Briefly explain how you obtained the x, y, and z coordinates of the arguments of the function. How does `Haskell` refer to this technique?

Question 1.b. (10 marks)

This question is about a user-defined implementation of a binary tree.

- A tree is either a leaf node or an internal node;
- Each internal node stores a character and has a left and a right child, which are also trees;
- A leaf node does not store any data.

Using record syntax, provide a data definition for the binary tree. Provide two good reasons why your definition is better than a definition that doesn't use record syntax.

Question 2: List Processing. (20/80 marks)

Question 2.a. (10 marks)

The module `Data.List` defines a function called `transpose` that transposes the rows and columns of a list consisting of lists. For example,

- `transpose [[1,2],[3,4]] = [[1,3],[4,2]];`
- `transpose [[1,2,3],[4,5,6]] = [[1,4],[2,5],[3,6]];`
- `transpose [[1,2,3],[4,5,6],[7,8,9]] = [[1,4,7],[2,5,8],[3,6,9]];` ...

Provide an implementation of `transpose` for argument lists consisting of lists of arbitrary precision integers. You may assume all members of an argument list have the same length.

Question 2.b. (10 marks)

Define a function that implements the quicksort algorithm for a list consisting of arbitrary precision integers.

Question 3: Higher-Order Functions and Advanced Expressions.

(20/80 marks)

Some of the subquestions in this question are about the following function.

```
mapFuns :: a -> [a -> b] -> [b]
```

The function returns a list consisting of the application of the members of its second (list) argument to its first argument.

Question 3.a. *(2 marks)*

What is a higher-order function? There is no need to explain your answer.

Question 3.b. *(2 marks)*

Is the function `mapFuns` a higher-order function? Explain your answer.

Question 3.c. *(1 mark)*

What is a partial application? There is no need to explain your answer.

Question 3.d. *(3 marks)*

Provide an example of a partial application, state the type of the partial application, and provide a description of the semantics of the partial application.

Question 3.e. *(4 marks)*

Provide an implementation of `mapFuns` that uses `map` and an anonymous function.

Question 3.f. *(4 marks)*

Provide an implementation of `mapFuns` that uses `map` and an operator section that uses the (function) application operator.

Question 3.g. *(4 marks)*

Using standard `Haskell` functions (only) provide an implementation of `mapFuns` that uses a partial application. The implementation does not have to be point-free.

Question 4: Types and Type Classes.

(20/80 marks)

(14 marks)

Question 4.a.

Provide a user-defined, polymorphic class for two-dimensional coordinates. The name of the class should be `Coordinate`. The class should define:

- A function called `createCoordinate` for creating an instance of the class;
- A function called `getFirst` for getting the first coordinate of an instance of the class;
- A function called `getSecond` for getting the second coordinate of an instance of the class; and
- A function called `addCoordinates` for *adding* two instances of the class; Here two instances are added by pairwise adding their corresponding coordinates.

(6 marks)

Question 4.b.

Provide an implementation of the `Coordinate` class from the previous question for instances of `Haskell`'s built-in pair class.