Question 1: Basics. (20/70 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)

A book is either a hardback or an audio book. Each book has a price attribute. A hardback has a page count attribute and an audio book has a duration attribute.

- Using record-syntax, provide a user-defined data type for a book.
- State *one* advantage of the record-syntax.

Question 2: List Processing. (15/70 marks)

Question 2.a. (5 marks)

A triple (x, y, z) of *positive* integers is *Pythagorean* if $x^2 + y^2 = z^2$. Define a function `pyths` that returns the Pythagorean triples whose components do not exceed the argument to the function. E.g. `pyths 10` returns `[(3,4,5),(4,3,5),(6,8,10),(8,6,10)]`

Question 2.b. (5 marks)

The *polymorphic* library function `cycle` takes a list argument and turns it into an infinite cyclic representation of the list. E.g. `cycle [1,2]` returns `[1,2,1,2,1,2,...]`. Provide a recursive definition of `cycle`.

Question 2.c. (5 marks)

The *polymorphic* function `replicate` takes an `Int`, n, and an element and returns a list consisting of n such elements. Provide a non-recursive implementation of `replicate`. Explain your answer. (Your solution should not rely on list comprehensions or the functions defined in the `Enum` class.)

Question 3: Higher-Order Functions and Advanced Expressions. (20/70 marks)

Question 3.a. (2 marks)
What is a higher-order function?

Question 3.b. (2 marks)
State the name and the type of an existing higher-order function. (You will need the function for the next subquestion.)

Question 3.c. (2 marks)
Explain how the type of your previous answer demonstrates the function is a higher-order function.

Question 3.d. (2 marks)
Provide an example of a partial application. (You will need the example for the next two subquestions.)

Question 3.e. (2 marks)
What is the type of your partial application?

Question 3.f. (2 marks)
What are the semantics of your partial application?

Question 3.g. (2 marks)
Provide an example of an application of an operator section application to a `Bool` argument. What is the result?

Question 3.h. (3 marks)
The function `square_of_successor` returns the square of the successor of its argument. Complete the following definition of the function. You are not allowed to use a lambda expression or auxiliary function definitions. *Hint: use composition.*

```
square_of_successor ::
square_of_successor =
```

Question 3.i. (3 marks)
Provide an expression that returns the infinite list `[1,-2,3,-4,...]`. There is no need to provide a signature.

Question 4: Types and Type Classes. (15/70 marks)

Question 4.a. (10 marks)
The class `Comparable` extends the `Eq` class and provides the following functions for comparing numbers: `le` (less than or equal), `lt` (less than), `ge` (greater than or equal), and `gt` (greater than). Provide a definition for the class `Comparable`: it should require minimum effort for the user to define instances of the class.

Question 4.b. (5 marks)
Provide an implementation of the `Comparable` class from the previous question for instances of `Haskell`'s built-in `Bool` type. The value `True` should be the maximum value.