1. **Wie hoch ist der Aufwand um Zusicherungen im Eiffel-Code zu formulieren?**
    - **How hard is it to implement contracts in Eiffel?**

The effort is low, since contracts in Eiffel are integrated as a part of the programming language. A Developer can define preconditions, postconditions and class invariants without adding additional Librarys. A precondition clause is introduced by the keyword require, a postcondition clause by the keyword ensure and invariants by the keyword invariant.

2. **Wie stark wirkt sich die Überprüfung von Zusicherungen auf die Laufzeit aus?**
    - **How big is the influence of the contracts monitoring for the run-time?**

The monitoring of the contracts is influencing the run-time. The compiler allows you to set the compilation options seperately for each class:

- no : assertions have no run-time effect.
- require : monitor preconditions only, on routine entry.
- ensure : preconditions on entry, postconditions on exit.
- invariant : same as ensure, plus class invariant on both entry and exit for qualified calls.
- all : same as invariant, plus check instructions, loop invariants and loop variants.

No and require are recommended, but the best solution is to use the restrictions according to the individual needs.

The Eiffel book recommends to activate contracts only during the development, hence they are mainly used for finding bugs (https://docs.eiffel.com/book/method/et-design-contract-tm-assertions-and-exceptions)

3. **Vorbedingungen dürfen im Untertyp nicht stärker und Nachbedingungen nicht schwächer werden um Ersetzbarkeit zu garantieren. Der Eiffel-Compiler überprüft diese Bedingungen. Ist es (trotz eingeschalteter Überprüfung von Zusicherungen) möglich, diese Bedingungen zu umgehen? Wenn ja, wie?**
    - **Preconditions are not allowed to be stronger in the subtype and postconditions are not allowed to be weaker, so that it is possible to ensure the substitutability. The Eiffel Compiler checks those conditions. Is it (despite the activated monitoring of the contracts) possible to avoid those conditions? If yes, how?**

When you are overwriting a method of a subtype, you can change the precondition only with require else (make the precondition only weacker) and the postcondition only with ensure then (make the postcondition stronger).

This can be avoided with using constants for the conditions and setting other values in the subtype, so that the conditions can get stronger or weaker

4. **Eiffel erlaubt kovariante Eingangsparametertypen. Unter welchen Bedingungen führt das zu Problemen, und wie äußern sich diese? Schreiben Sie ein**

Programm, in dem die Verwendung kovarianter Eingangsparametertypen zu einer Exception führt.

Type errors (called catcalls in Eiffel speak) are possible. The compiler cannot detect these type errors. They usually trigger an exception at runtime.
These type errors are possible due to covariant redefinition of arguments and polymorphy. Both principles are very powerful in OO programming. Other languagues (like java, scala, etc.) solve this problem by disallowing covariant redefinitions of arguments and keeping polymorphy.

5. **Vereinfachen kovariante Eingangsparametertypen die Programmierung? Unter welchen Bedingungen ist das so?**
    - **Are covariant input parameter types simplifying the programming? Under which conditions is it so?**

Covariant input parameters are usefull in cases when realistic scenarios are programmed. For example in our exercise:
STUDENT is a subtype of person. The Class ACCOUNT defines a method set_owner, which expects a Object of Type Person as Parameter.ACCOUNT _STUDENT is a subtype of in ACCOUNT. Of course it makes more sense that ACCOUNT_STUDENT acepts only STUDENT in the method set_owner.

6. **Welche Spracheigenschaften von Eiffel finden Sie interessant und würden Sie gerne auch in anderen Sprachen sehen? Welche Eigenschaften von Eiffel empfinden Sie dagegen als störend?**