|  |
| --- |
| Question 1:  Define type rules in the context of functional programming. How do type rules contribute to program correctness and safety? Provide programming examples to illustrate your explanation. |
| Type rules exist within a type system, where it ensures that variables, expressions, and functions are used consistently and safely.  It can catch errors early in the development process, improves code readability and program reliability.  It can guarantee the absence of behaviours such as the existence of a method/field when invoked on a particular object.  It enforces higher-level modularity properties such as maintaining integrity of data abstractions and check for violation of information hiding.  It also enforces disciplined programming. It forms the backbone of module-based languages for large-scale composition types, encouraging abstract design. |

|  |
| --- |
| Question 2:  Discuss the differences between static typing and dynamic typing in functional programming. How do type rules manifest in each of these typing systems? Provide examples and analyze the advantages and disadvantages of each approach. |
| Static typing:   * Variables types determined at compile time * Cannot change during execution * Examples: Java, C, C++ * C++: Data types explicitly defined   Dynamic typing:   * Variable types determined at runtime * Can change during execution * Examples: Python, JavaScript, Ruby   Statically typed languages have better performance (compile-time optimizations)  Dynamic typing: ease of use, conciseness (do not need to specify type explicitly)  Error detection:  Static: compiler catches type-related errors – more reliable  Dynamic: Detect type issues at runtime (unexpected type errors)  Performance:  S: Optimised by compiler  D: Slightly slower execution (runtime type checks)  Ease of Use:  S: Additional type annotations – more verbose code  D: Simplicity – concise code without explicitly declaring data types  Flexibility:  D: variables can change during runtime – data type not known yet is beneficial  Type safety:  S: Higher level type safety – type-related error at compile-time  D: sacrifices some type safety for flexibility & ease of use  <https://www.bairesdev.com/blog/static-vs-dynamic-typing/> |

|  |
| --- |
| Question 3:  Explain the concept of type inference in functional programming languages. How does type inference work, and what role do type rules play in this process? Provide examples to demonstrate type inference in action |
| Compiler automatically deduce the types of expression & variables  Without explicit type annotations from the programmer  Maintains verbosity while maintaining type safety  Example:  Function Foo(a,b) = x+y,  Compiler knows “+” takes 2 integers and also returns an integer.  Thus, it is inferred that Foo has type integer.  <https://www.techopedia.com/definition/22695/type-inference#:~:text=Techopedia%20Explains%20Type%20Inference&text=As%20a%20basic%20example%2C%20consider,also%20has%20the%20type%20integer> |

|  |
| --- |
| Question 4:  Explore the concept of parametric polymorphism in the context of type rules. Provide a detailed explanation of parametric polymorphism, including its benefits and how it is enforced by type rules. Offer examples to illustrate your points |
| Enables generic definition of functions & types without concern for type-based errors  Benefits: More expressive while writing generic code   * Allows programmers to reuse, evaluate and execute the program * Reduce pairing of multiple functionalities * Method overloading 🡪 multiple ways to init class objs * Method overriding 🡪 Code reuse of existing groups * Faster coding at runtime   Example: Joining a list of integers with a floating point value  <https://www.techopedia.com/definition/21247/parametric-polymorphism#:~:text=Parametric%20polymorphism%20is%20a%20programming,to%20various%20types%20of%20data>  <https://teachcomputerscience.com/polymorphism/#:~:text=Parametric%20polymorphism%20enables%20programmers%20to,data%20in%20the%20same%20way> |

|  |
| --- |
| Question 5:  Investigate the role of algebraic data types (ADTs) in enforcing type rules in functional programming languages. Discuss how ADTs contribute to type safety and provide examples demonstrating their usage. |
|  |