

Exam Questions

Part 1. λ -calculus.

1. Let us define a language by the following BNF grammar:

$t ::= v \mid t \& t \mid t \% t \mid (t)$

$v ::= a \mid b \mid c \mid d$

Determine which of the following expressions belong to this language:

$a \& b$

$(b \& c) (a \% d)$

$d \% (a \% (b \& c))$

$a \& (b (c))$

$((a) \& (b \% c)) \& (d \% d)$

Explain how you arrived at your answer.

2. Evaluate the following λ -expressions to a value:

$(\lambda x. x + 3) 12$

$(\lambda x. (x + (\lambda x. x + x) 4 + 10)) 3$

$(\lambda y. y) (\lambda z. z + 3)$

$(\lambda z. z 10) (\lambda z. z + (z + 4))$

$(\lambda a. \lambda b. a (b b) 12) (\lambda x. \lambda y. x (y + 2)) (\lambda x. x)$

3. Explain the meaning of the evaluation rule:

$$\frac{t_1 \rightarrow t_1'}{t_1 \text{ and } t_2 \rightarrow t_1' \text{ and } t_2}$$

4. What does it mean that evaluation of an expression is *stuck*?
5. For the language of λ -calculus that we used in the course, give three examples of expressions whose evaluation will get stuck.
6. Give an example of two evaluation rules that, when present in the same system, result in a non-deterministic evaluation.
7. Give an example of evaluation rules resulting in a non-confluent evaluation.
8. Can a non-confluent evaluation be deterministic?
9. Can deterministic evaluation be non-confluent?
10. Write the evaluation rule for β -reduction.
11. What does it mean that two expressions are equivalent “up to α -conversion”? Give an example.
12. Write evaluation rule(s) for call-by-value evaluation of function application.
13. Write evaluation rule(s) for call-by-name evaluation of function application.
14. What is the difference between call-by-name and call-by-need evaluation?
15. Write evaluation rule(s) for strict function application.
16. Write evaluation rule(s) for lazy function application.

17. Write evaluation rule(s) for strict AND.

18. Write evaluation rule(s) for lazy AND.

Part 2. Programming languages and type systems.

19. Give at least three important differences between imperative and functional programming paradigm.

20. Can a language combine functional and imperative programming features? How?

21. What is a type system?

22. What is the purpose of a type system?

23. Explain the meaning of the following typing rule:

$$\frac{t_1 : T \quad t_2 : T}{(t_1 \text{ and } t_2) : T}$$

24. What does it mean that a term is syntactically correct?

25. What does it mean that a term is well-typed?

26. Describe type derivation for the expression:
 $(\lambda x : \text{Int} \rightarrow \text{Int}. x (5 + x 7)) (\lambda y : \text{Int}. y - 3)$

27. What is the relationship between typing rules and evaluation rules?

28. Can a type system disallow a term that would evaluate without being stuck? If so, give an example.

29. What properties constitute soundness of a type system? Explain each of them.

30. Explain the notions of explicit and implicit typing. Give examples of languages.

31. Explain the notions of static and dynamic typing. Give examples of languages.

32. What is a record (struct) type? Give examples in Timber and in Java.

33. What is a data (enum) type? Give examples in Timber and in Java.

34. What is type coercion? You can limit the reply to the case of coercion of a term of a certain type T1 to its supertype T2.

35. What does it mean that a type T1 is a subtype of the type T2? Give examples for subtyping/supertyping of record (struct) and data types.

36. How does subtyping normally works for function types?

37. What is a polymorphic function? Give an example of such a function. Explain the advantages of writing polymorphic code.

38. What is a type class? How can it be used? Give an example.

39. Discuss implementation of type classes using witnesses. Give an example.

Part 3. Object-oriented programming.

40. Name the main ideas of object-oriented programming.
41. What is an object in OO programming?
42. What is a class in OO programming?
43. Explain the idea of state encapsulation. Do Java and / or Timber offer it?
44. Explain the idea of multiple representations (i.e. separation of interface from implementation). Do Java and / or Timber offer it?
45. What is the difference between structural and nominative (or declarative) subtyping?
46. Why is subtyping most useful together with implicit type coercion? Explain using an example.
47. What is inheritance (subclassing)? What is the difference between subtyping and subclassing?
48. Explain the difference between subclassing by delegation and subclassing by embedding.
49. What is method overriding and how can it be implemented using method tables?
50. Explain the difference between static and dynamic dispatch (method lookup). Discuss the implementation techniques for each of them.
51. Explain the terms “open recursion” / “late binding of *this*” in OO programming.

Part 4 . Parallelism and concurrency.

52. Explain the difference between parallelism and concurrency.
53. Define the two performance metrics: throughput and responsiveness. How can they be improved on a single-core machine? on a multi-core / multi-cpu machine?
54. Define task-level parallelism.
55. Define thread-level parallelism.
56. Define instruction-level parallelism.
57. Explain the notion of interleaving execution (of tasks or threads).
58. Name at least two problems with using threads.
59. Give examples of visible and transparent parallelism.
60. Explain the notions of execution threads, thread synchronization, locks (mutexes and semaphores).
61. Why is thread synchronization important to protect memory consistency?
62. Explain the scheduling principles of tinyOS. What is the difference between an event and a task?

63. Name the different states that a task in Ada can be in. Why can a task be blocked?
64. Do we need synchronization or locking mechanisms in a time-triggered language like Giotto? Why?
65. What possibilities for program parallelization do functional languages offer?
66. Explain the idea of object-level concurrency. What is the difference between active and reactive objects?

Part 5. System design.

67. Define the terms correctness, reliability, robustness, maintainability.
68. What constitutes a software design methodology?
69. What are the main weaknesses of V- and Waterfall design models?
70. What is a model of a software system and what can it be used for?
71. What is a modeling language?
72. What is an interface description language (IDL)?
73. What is a distributed software system?
74. What is an embedded software system? What are the typical properties of an embedded system?
75. What is a hard/soft real-time system?
76. Give a definition of a software component.
77. What are the advantages of component-based software design?
78. What are the challenges of component-based design of embedded software systems?
79. What is the difference between components and services?