Names, Scopes, and Bindings

1. What is binding time?

2. Explain the distinction between decisions that are bound statically and those that are bound dynamically.

3. What is the advantage of binding things as early as possible? What is the advantage of delaying bindings?

4. Explain the distinction between the lifetime of a name-to-object binding and its visibility.

5. What determines whether an object is allocated statically, on the stack, or in the heap?

6. List the objects and information commonly found in a stack frame.

7. What is a frame pointer? What is it used for?

8. What is a calling sequence?

9. What are internal and external fragmentation?

10. What is garbage collection?

11. What is a dangling reference?

12. What do we mean by the scope of a name-to-object binding?

13. Describe the difference between static and dynamic scoping.

14. What is elaboration?

15. What is a referencing environment?

16. Explain the closest nested scope rule.

17. What is the purpose of a scope resolution operator?

18. What is a static chain? What is it used for?

19. What are forward references? Why are they prohibited or restricted in many programming languages?

20. Explain the difference between a declaration and a definition. Why is the distinction important?

21. Explain the importance of information hiding.

22. What is an opaque export?

23. Why might it be useful to distinguish between the header and the body of a module?

24. What does it mean for a scope to be closed?

25. Explain the distinction between "modules as managers" and "modules as types."

26. How do classes differ from modules?

27. Why might it be useful to have modules and classes in the same language?

28. Why does the use of dynamic scoping imply the need for run-time type checking?

29. Explain the purpose of a compiler's symbol table.

30. What are aliases? why are they considered a problem in language design and implementation?

31. Explain the value of the `restrict` qualifier in C.

32. What is overloading? How does it differ from coercion and polymorphism?

33. What are type classes in Haskell? What purpose do they serve?

34. Descibe the difference between deep and shallow binding of referencing environments.

35. Why are binding rules particlarly important for langauges with dynamic scoping?

36. What are first-class subroutines? what languages support them?

37. What is a subroutine closure? What is it used for? How is it implemented?

38. What is an object closure? How is it relatedto a subroutine closure?

39. Describe how the delegates of C# extend and unify both subroutine and object closures.

40. Explain the distinction between limited and unlimited extent of objects in a local scope.

41. What is a lambda expression? How does the support for lambda expressions in functional languages compare to that of C# or Ruby? To that of C++ or Java?

42. What are macros? What was the motivation for including them in C? What problems may they cause?

43. List the basic operations provided by a symbol table.

44. Outline the implementation of a LeBlanc-Cook style symbol table.

45. Why don't compilers generally remove names from the symbol table at the ends of their scopes?

46. Describe the association list (A-list) and central reference table data structures used to implement dynamic scoping. Summarize the tradeoffs between them.

47. Explain how to implement deep binding by capturing the referencing environment A-list in a closure. Why are closures harder to build with a central reference table?

48. What purpose(s) does separate compilation serve?

49. What does it mean for an external variable to be linked in C?

50. Summarize the C convention for use of `.h` and `.c` files.

51. Describe the difference between a compilation unit and a C++ or C# namespace.

52. Explain why Ada and similar languages separate the header of a module from its body. Explain how Java and C# get by without.