Cotrol Flow

1. Nane eight major categories of control-flow mechanisms.

2. What distinguishes operators from other sorts of functions?

3. Explain the difference between prefix, infix, and postfix notation. What is Cambridge Polisg notation? Name two programming languages that use postfix notation.

4. Why don't issues of associativity and precedence arise in Postscript or Forth?

5. What does it mean for an expression to be referentially transparent?

6. What is the difference between a value mode of variables and a reference model of variables? Why is the distinction important?

7. What is an l-value? An r-value?

8. Why is the distinction between mutable and immutable values important in the implementation of a language with a reference model of variables?

9. Define orthogonaality in the context of a programming language design.

10. What is the difference between a statement and an expression? What does it mean for a language to be expression-oriented?

11. What are the advantages of updating a variable with an assignment operator, rather than with a regular assignment in which the variable appears on both the left-hand side and right-hand side?

12. Given the ability to assign a value into a variable, wy is it useful to be able to specify an initial value?

13. What are aggregates? Why are they useful?

14. Explain the notion of definite assignment in Java and C#.

15. Why is it generally expensive to catch all uses of uninitialized variables at run time?

16. Why is it impossible to catch all uninitialized variables at compile time?

17. Why do most languages leave unspecified the order in which the arguments of an operator or function are evaluated?

18. What is short-circuit Boolean evaluation? Why is it useful?

19. List the principal uses of `goto`, and the structured alternatives to each.

20. Explain the distinction between exceptions and multilevel returns.

21.What are continuations? What other language features do they subsume?

22. Why is sequencing a comparatively unimportant form of control flow in Lisp?

23. Explain why it may sometimes be useful for a function to have side effects.

24. Describe the `jump` code implementation of short-circuit Boolean evaluation.

25. Why do imperative languages commonly provide a `case` or `switch` statement in addition to `if...then...else`?

26. Describe three different search strategies that might be employed in the implementation of a `case` statement, and the circumstances in which each would be desirable.

27. Explain the use of `break` to terminate the arms of a C `switch` statement, and the behavior that arises if a `break` is accidentally omitted.

28. Describe three subtleties in the implementation of enumeration-controlled loops.

29. Why do most languages not allow the bounds of increment of an enumeration-controlled loop to be floating-point numbers?

30. Why do many languages require the step size of an enumeration-controlled loop to be a compile-time constant?

31. Describe the ”iteration count” loop implementation. What problem(s) does it solve?

32. What are the advantages of making an index variable local to the loop it controls?

33. Does C have enumerations-controlled loops? Explain.

34. What is a collection (a container instance)?

35. Explain the difference between true iterators and iterator objects.

36. Cite two advantages of iterator objects over the use of programming conventions in a language like C.

37. Describe the approach to iteration typically employed in languages with first-class functions.

38. Given an example in which a mid-test loop results in more elegant code than does a pretest or post-test loop.

39. What is a tail-recursive function? Why is tail recursion important?

40. Explain the difference between applicative-order and normal-order evaluation of expressions. Under what circumstances is each desirable?

41. What is lazy evaluation? What are promises? What is memoization?

42. Give two reasons why lazy evaluation may be desirable.

43. Name a language in which parameters are always evaluated lazily.

44. give two reasons why a prorammer might sometimes want control flow to be nondeterministic.

45. Explain how Icon generators differ from the iterators of Clu, Python, Ruby, and C#, and from the iterator objects of Euclid, C++, and Java.

46. Describe the notations of success and failure in Icon.

47. What is backtracking? Why is it useful?

48. Name a language other than Icon n which backtracking plays a fundamental role.

49. What is a guarded command?

50. Explain why nondeterminacy is particularly important for concurrent programs.

51. Give three alternative definitions of fairness in the context of nondeterminacy.

52. Desrie three possible ways of implementing the choice among guards that evaluate to true. What are the tradeoffs among these?