Review questions for midterm

1. What is a regular expression? What is a DFA/NFA? (See questions on the homework relating to these concepts.)

2. What is a context free grammar? (Again, see homework questions.)

3. What is a LL grammar? What is an LR grammar? Why are they useful classes to consider? (Hint: see question 8 below.)

4. Be able to construct FIRST and FOLLOW sets for an LL grammar, and the predictive parsing table. (See examples from class, as well as the homework.)

5. What is a leftmost versus a rightmost derivation? Be able to construct these, as well as parse trees.

6. What is lex/flex? (Recall basic rules, etc.)

7. What is the goal of tokenizing? Why is it done separately from parsing?

8. What is the big-O complexity of parsing? (Hint: there isn't one answer to this question! Be able to explain those trade-offs we discussed. This should also connect to your answer to question 3, by the way.)

9. What is binding time?

10. What is the advantage of binding as early as possible? Why do we delay bindings despite this in some languages? Give an example of a language that does each.

11. What do lifetime and visibility mean in the context of binding?

12. What do we mean by the scope of a binding?

13. What is dynamic scoping (versus static scoping)?

14. What is elaboration?

15. What is a closed scope?

16. What does the use of dynamic scoping imply the need for run-time type checking? Give an example of a language again, and explain specific features that this choice gives (or takes away from) the language.

17. Explain the differences (and similarities) between overloading, coercion, and polymorphism. Give an example of each of them in a programming language.

18. What is the difference between the reference model of variables and the value model? Why is the distinction so important? Give examples of languages that use each model.

19. Why is the distinction between mutable and immutable variables so important in a language with a reference model for variables? (Hint: Think Python.)

20. List a few of the main uses for goto, and the (much better!!) alternatives for each.

21. What is a continuation?

22. Why do languages provide case statements if they already have if-then-else statements?

23. Why do many languages require that the step size of an enumeration controlled loop be a compile-time constant? In a similar vein, why do languages not allow the bounds of increment of an enumeration controlled loop to be floating-point numbers? What do you lose (and gain) in the language if you relax these requirements?

24. What are the advantages and disadvantages of making the index variable local to the loop it controls?

25. What is tail-recursion, and why is it important?

26. What is lazy evaluation? When is it used, and why? Give an example of a language where it is used.

27. Define and compare/contrast the following: strongly typed, statically typed, dynamically typed, weakly typed. Be ready with some examples, too.

28. Give two examples of languages that lack a boolean type. What do they use instead? What are some advantages and disadvantages of this model?

29. What is the difference between type equivalence and type compatibility?

30. What are structural and name equivalence? Discuss comparative advantages, and give examples of languages that use each.

31. What are strict and loose name equivalence?

32. Under what circumstances does a type conversion require a run-time check? Give at least 2 examples.

33. What are "holes" in records? When do they arise, and what are the strategies that are used to minimize the problems they cause?

34. Under what circumstances can an array declared within a subroutine be allocated in the stack? Under what circumstances can it be allocated in the heap?

35. What is the difference between row-major and column-major layouts of arrays, and why should a programmer care which one is used?

36. What are dangling references? When are they created, and why are they a problem?

37. What is garbage collection? Describe the reference count model of garbage collecting, and explain briefly why it doesn't always work.

38. Why was automatic garbage collection so slow to be adopted by modern programming languages? (In other words, what are the disadvantages of automatic garbage collection?)

39. Describe the strategies we discussed to mitigate dangling references (tombstones and locks and keys), and how they solve the problem. Are there any weaknesses associated with them?

40. Most statically typed languages since the 1970s (like Java and C#) use some form of name equivalence. Is structural equivalence a bad idea? Why or why not?

41. What type of information is added to the stack within a frame when a new function is called?