FINAL EXAM: Tuesday, 25 April 2017, 2pm, Room 3342b

NAME AS APPEARS ON STUDENT ID:

STUDENT ID NUMBER:

GAUL/CONFLUENCE USER NAME:

REMINDERS (from course outline):

1. The final exam will be closed book, closed notes, with no electronic devices allowed, with particular reference to any electronic devices that are capable of communication and/or storing information.

1. In Scala, the type that every type is a subtype of is called ANSWER.
   ANSWER=

2. In Haskell, we can declare the type of a parameter to a function to be something specific like Char. However, we can also declare the type of a parameter to be something that could include many types like ListLike that supports the functions head and tail. We do this with a definition of ListLike that begins with the keyword ANSWER.
   ANSWER=

3. In the Ruby community, the acronym DSL is an abbreviation for ANSWER.
   ANSWER=

4. ANSWER would be the derivation of ((1)) in the language defined by the context-free grammar consisting of the two rules $E \rightarrow (E)$ and $E \rightarrow 1$.
   ANSWER=

5. In Ruby, by convention, the ? in the method me? is used to indicate that me is ANSWER.
   ANSWER=

6. In the ICD textbook's example interpreter for evaluating expressions, in the row labelled id, we have the code: v = lookup(vtable, getname(id)) ; if v = unbound then error() else v. The value of v would be unbound in the situation that ANSWER.
   ANSWER=

7. Three concepts related to concurrency were discussed with regards to the language Io. ANSWER was presented as a way to manage two execution streams that pass control back and forth between themselves.
   ANSWER=

8. The specific type of context-free grammar that was the main focus of the portion of the Syntax Analysis chapter that was assigned was ANSWER.
   ANSWER=

9. Unlike most Lisp systems, Clojure doesn't use its own custom virtual machine. It was originally designed to compile to code that would run on the ANSWER.
   ANSWER=

10. Three concepts related to concurrency were discussed with regards to the language Io. ANSWER was presented as a general mechanism for sending a message to an object that would cause that object to respond to the message as a separate process running asynchronously.
    ANSWER=

11. With respect to the value returned by the Ruby expression 'hi'.object_id == 'hi'.object_id, you can say it ANSWER.
    ANSWER=

12. In Ruby, if you declare a class with a class name that is already in use and put in it the definition of a new method, you have changed the functionality of the existing class (even if it is a predefined class like Fixnum). The property of Ruby that allows this is ANSWER.
    ANSWER=

13. In the context-free grammar $A \rightarrow BA$, $B \rightarrow AB$, $A \rightarrow B$, $A \rightarrow a$, $B \rightarrow b$, and $B \rightarrow$ the value of FIRST(A) is ANSWER.
    ANSWER=

14. If the types of a programming language are bound at execution time rather than compile time, then the types are called ANSWER.
    ANSWER=

15. A compiler typically keeps track of which names are associated with which objects by using ANSWER.
ANSWER=

16. The root of the inheritance hierarchy in Ruby is the class ANSWER.
ANSWER=

17. Programming languages that view programming as describing a step-by-step process to do something are called ANSWER languages.
ANSWER=

18. Using the straightfoward statement translation scheme in the textbook, if I were to TransStat('while z $<$ 3 do z := 1 + z', vtable, ftable), newvar() be invoked ANSWER times.
ANSWER=

19. In Haskell's do notation for working with monads, assignment uses the ANSWER operator.
ANSWER=

20. The feature of programs being able to 'write programs' (creating application specific language features) is called ANSWER.
ANSWER=

21. The loop and recur constructs are in Clojure to guide ANSWER.
ANSWER=

22. In Haskell, $[x * 2 | x \leftarrow [3, 4, 5]]$ evaluates to ANSWER.
ANSWER=

23. In Haskell, the type signature of the function sum x y = x + y is ANSWER.
ANSWER=

24. One of the three most significant parts of a Haskell monad is called ANSWER, which unwraps a function.
ANSWER=

25. The portion of the call stack associated with a single function invocation and running is called ANSWER.
ANSWER=

26. In Ruby, normally, when you try to add a String to a Fixnum, you get an error message saying that a String can't be coerced to a Fixnum. This is because Ruby is ANSWER typed.
ANSWER=

27. In Erlang, you can link two processes together. Then when one dies, it sends ANSWER to its twin.
ANSWER=

28. As one would expect in an object-oriented language, when a message is sent to an object, the first thing the system does is to look for a corresponding method in that object. However, Io lets you change what happens next by redefining the method named ANSWER.
ANSWER=

29. Instead of the + symbol, Haskell uses the symbol ANSWER for a string concatenation operator.
ANSWER=

30. Type checking done during program execution is called ANSWER.
ANSWER=

31. In Prolog, the most natural way to express the query 'what animals are cats?' is ANSWER.
ANSWER=

32. Scala uses few type declarations because its compiler does ANSWER.
ANSWER=

33. In Clojure, you cannot change a reference outside of ANSWER.
ANSWER=

34. In Prolog, the expression $[1, 2, 3] = [X|Y]$ causes Y to have the value ANSWER.
ANSWER=

35. Io allows programmers to play with its syntax, doing things like introducing a colon operator and redefining how curly braces are processed. This makes it easy to use Io to create ANSWER.
ANSWER=

36. In Haskell, defining lists using a notation like $[x * 2|x \leftarrow [3, 4, 5]]$ is called using ANSWER.
ANSWER=

37. In the ICD textbook's example interpreter for evaluating expressions, in the row labelled id, we have the code: v = lookup(vtable, getname(id)) ; if v = unbound then error() else v. It says getname(id) instead of id, because ANSWER.
ANSWER=

38. When a grammar can produce two distinct syntax trees for the same string, the grammar is said to be ANSWER.
ANSWER=

39. In Io, the basic method for creating a new object is ANSWER.
ANSWER=

40. One aspect of the if-then-else-end syntax of Ruby is that it avoids the ANSWER problem.
ANSWER=

41. The Scala feature closest to a Ruby mixin is the ANSWER.
ANSWER=

42. With respect to the value returned by the Ruby expression :hi.object_id == :hi.object_id, you can say it ANSWER.
ANSWER=

43. In Haskell, ANSWER is the output of zip [17..20] [10,8..4].
ANSWER=

44. Adding a feature to a programming language to make it easier to do something that was already doable is called adding ANSWER.
ANSWER=

45. In Prolog, the expression $X = [[1, 2]|[3, 4]]$ causes X to have the value ANSWER.
ANSWER=

46. When you write a parser for a context-free grammar that satisfies the LL(1) criteria by representing each non-terminal by a function that chooses what functions to invoke by the LL(1) criteria, this sort of parser is called ANSWER.
ANSWER=

47. Some languages allow a function to be ANSWER, that is to be defined over a large class of similar types, e.g., over arrays no matter what type their elements are.
ANSWER=

48. ANSWER means that the language allows the same name to be used for different operations over different types.
ANSWER=

49. The specifications for how to group characters into meaningful units are traditionally written as ANSWER.
ANSWER=

50. In Haskell, given the definition sum x y = x + y, ANSWER is the value of that is produced by the expression (sum 3).
ANSWER=

51. Matz, the creator of Ruby, thinks that it is less important to optimize the execution (efficiency) of a programming language and more important to optimize the efficiency of ANSWER.
ANSWER=

52. In a context-free grammar, the nonterminal that derives an entire member of the language being defined is called ANSWER.
ANSWER=

53. In automatically generating the code that reads characters and outputs the part of a programming language that is analogous to its words, we start with a specification and then traditionally convert it into code in two stages. In the first stage, we produce ANSWER.
ANSWER=

54. Since a compiler may have to look up what object is associated with a name many times, it is typical to use ANSWER to avoid linear search times.
ANSWER=

55. Many programming languages represent internal constants for types like strings, floats, and integers. Scala has the unusual distinction of having an internal constant representation for the type ANSWER, which is normally viewed as a format external to a program.
ANSWER=

56. In the chapter on Scala, we get the following interesting quote: ANSWER is the most important thing you can do to improve code design for concurrency.
ANSWER=

57. Another method of parameter passing, whose technical name is ANSWER, is implemented by passing the address of the variable (or whatever the given parameter is). Assigning to such a parameter would then change the value stored at the address.
ANSWER=

58. In Prolog, the most natural way to express the rule that 'I am an ancestor of you if I am a parent of you' is ANSWER.
ANSWER=

59. In Clojure, ANSWER is a concurrency construct that allows an asynchronous return before computation is complete.
ANSWER=

60. A major claim in object-oriented design philosophy is that you should code to ANSWER rather than code to implementation.
ANSWER=

61. In Haskell, if I write (h:t) = [3, 5, 7], ANSWER is the value of h.
ANSWER=

62. The command name for the Ruby interpreter is ANSWER.
ANSWER=

63. Type checking done during program compilation is called ANSWER.
ANSWER=

64. Since Haskell doesn't have traditional error handling, by convention, people use the ANSWER monad to distinguish a valid return from an error return.
ANSWER=

65. In C, when you pass a function as a parameter to another function, it is implemented as passing ANSWER.
ANSWER=

66. In Io, we create a singleton by redefining the method ANSWER.
ANSWER=

67. One of the three most significant parts of a monad is called ANSWER, which wraps up a function and puts it in the monad's container.
ANSWER=

68. In Prolog, the expression hi(X, 4) = hi(3, Y) causes X to have the value ANSWER.
ANSWER=

69. In the context-free grammar $A \rightarrow BA$, $B \rightarrow AB$, $A \rightarrow B$, $A \rightarrow a$, $B \rightarrow b$, and $B \rightarrow$ the value of Nullable(A) is ANSWER.
ANSWER=

70. The technical term for the compiler design methodology where the translation closely follows the syntax of the language is ANSWER.
ANSWER=

71. Using the context-free grammar based on the two rules $A \rightarrow bA$ and $A \rightarrow b$, ANSWER would be the derivation sequence for bbb.
ANSWER=

72. To execute a code block in Ruby that is passed to a method on its parameter list, you send that parameter the method ANSWER.
ANSWER=

73. ANSWER is the data structure used in language translation to track the binding of variables and functions to their type.
ANSWER=

74. In Io, the type of an object is generally the nearest ancestor that ANSWER.
ANSWER=

75. The different traversals of a syntax tree done during compilation associate information with the nodes of the tree. The technical term for this kind of information is ANSWER.
ANSWER=

76. One approach to speeding up an interpreter is to translate pieces of the code being interpreted directly into machine code during program execution, this is called ANSWER.
ANSWER=

77. In Haskell, instead of writing something like second x = head(tail(x)), you can write this without introducing the parameter x by using function composition. Doing that, you would write ANSWER.
ANSWER=

78. If the system stack is used for a call stack, then it becomes important for the caller to update the top of the stack before copying items into it. The reason is because we are worried about the top of the stack being changed by ANSWER after we have copied in information but before we updated the stack top.
ANSWER=

79. To execute a code block in Ruby that is passed to a method but doesn't appear on its parameter list, you use the keyword ANSWER.
ANSWER=

80. ANSWER is the regular expression that corresponds to the language defined by the context-free grammar with the three rules $A \to Aa$, $A \to Ab$, $A \to a$.
ANSWER=

81. Although Haskell is a statically typed language, we usually don't need to write type declarations because Haskell uses ANSWER to figure out what the types are.
ANSWER=

82. The central idea of context-free grammars is to define a language by productions. These productions say that a nonterminal symbol can be replaced by ANSWER.
ANSWER=

83. In Scala, the type that is a subtype of every type is called ANSWER.
ANSWER=

84. The main Clojure approach to concurrency is called ANSWER.
ANSWER=

85. In Ruby, you would group methods into a class. In Erlang, you group functions into ANSWER.
ANSWER=

86. The : notation in the Ruby expressions :hi is used to indicate that hi is ANSWER.
ANSWER=

87. When the structure of the syntax tree is used to determine which object corresponds to a name, this is called ANSWER.
ANSWER=

88. In Erlang, the main approach to concurrency is ANSWER.
ANSWER=

89. In Clojure, the value of (repeat 1) is ANSWER.
ANSWER=

90. ANSWER data structures have the property that there are operations on the structure can destroy or modify it.
ANSWER=

91. In Ruby, the expression Fixnum.class returns ANSWER.
ANSWER=

92. In discussing object-oriented languages, objects are organized into a class tree to support the property of ANSWER.
ANSWER=

93. In discussing object-oriented languages, being able to handle objects of related types is called ANSWER.
ANSWER=

94. In Ruby, true.class returns ANSWER.
ANSWER=

95. In the ICD textbook's example interpreter for evaluating expressions, in the row labelled let id = Exp1 in Exp2, we have the code: v1 = EvalExp(Exp1, vtable, ftable); vtableP = bind(vtable, getname(id), v1), EvalExp(Exp2, vtableP, ftable). The bind function changes vtable into vtableP by ANSWER.
ANSWER=

96. In Io, a message has three aspects that can be interrogated by the call method. They are: the sender, the reciever, and ANSWER.
ANSWER=

97. When you define a method in a class, normally it is meant to be invoked on an object of that class (an instance method). Sometimes it is meant to be invoked on the class name itself (a class method), like Date.parse('3rd Feb 2001'). In Ruby, to define a class method, we put ANSWER at the beginning of the method name in its definition.
ANSWER=

98. Scala was designed to connect two programming paradigms, which were ANSWER.
ANSWER=

99. The way Haskell handles functions with more than one parameter is called ANSWER.
ANSWER=

100. In Ruby, the @@ is used to indicate that the variable @@me is ANSWER.
ANSWER=

101. The application that caused a significant increase in the popularity of Ruby was a web framework called ANSWER.
ANSWER=

102. The specifications of how to group characters into meaningful basic units of a programming language are generlly implemented in code that has the abstract form of ANSWER.
ANSWER=

103. Many syntax features of Erlang, such as ending statements with a period, reflect the influence of the programming language ANSWER.
ANSWER=

104. ANSWER are two derivations of the string cc that produce distinct syntax trees from the context-free grammar $X \rightarrow XcY$, $Y \rightarrow X$, $Y \rightarrow$ and $X \rightarrow$.
ANSWER=

105. ANSWER data structures have the property that no operation on the structure will destroy or modify it.
ANSWER=

106. In Scala, if I want to redefine a method that is defined in my parent class, I indicate this by using the keyword ANSWER.
ANSWER=

107. The portion of the program where the name is visible is called its ANSWER.
ANSWER=

108. When viewed formally, a language is defined as a set of ANSWER.
ANSWER=

109. In the ICD textbook's example interpreter for evaluating expressions, in the row labelled id(Exps), we have the code: args = EvalExps(Exps,vtable,ftable). We pass ftable to EvalExps to handle ANSWER.
ANSWER=

110. In Ruby, the name of the method in the class Me that is automatically invoked when a new object of type Me is created with Me.new is ANSWER.
ANSWER=

111. In Clojure, (take 3 (iterate (fn [x] (* 2 x)) 2)) produces ANSWER.
ANSWER=

112. In Scala, to indicate that a variable is mutable, you introduce it with the ANSWER keyword.
ANSWER=

113. In Prolog, the expression 2 = 1 + X causes X to have the value ANSWER.
ANSWER=

114. In describing the properties of an object-oriented language, encapsulation means ANSWER.
ANSWER=

115. In Haskell, if we want to define a local named function inside a function definition, we use the keyword ANSWER.
ANSWER=

116. An unusual built-in constant construct in Erlang lets us write ¡¡4:3,1:3¿¿ to represent the value ANSWER.
ANSWER=

117. In Prolog, the most natural way to express the rule that 'I am an ancestor of you if I am a parent of an ancestor of you' is ANSWER.
ANSWER=

118. Using the straightfoward statement translation scheme in the textbook, if I were to TransStat('while true do z := 1 + z', vtable, ftable), newlabel() be invoked ANSWER times.
ANSWER=

119. The main concurrency approach used in Ruby is ANSWER.
ANSWER=

120. If I wanted to fix the grammar $E \rightarrow E + E$ and $E \rightarrow id$, so that it would only produce one syntax tree, which is left recursive; the new grammar would be ANSWER.
ANSWER=

121. In the context-free grammar $A \rightarrow BA$, $B \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$, $B \rightarrow$ the value of Nullable(A) is ANSWER.
ANSWER=

122. Io is known for taking ANSWER -based approach to object-oriented programming.
ANSWER=

123. The main programming paradigm in Erlang is ANSWER programming.
ANSWER=

124. When viewing programming languages as natural languages, the word ANSWER is used instead of 'words'.
ANSWER=

125. In the ICD textbook's example interpreter for evaluating expressions, in the row labelled id(Exps), we have the code: args = EvalExps(Exps,vtable,ftable). We pass vtable to EvalExps to handle ANSWER.
ANSWER=

126. The & notation in the line of Ruby def george(&sam) is used to indicate that sam is ANSWER.
ANSWER=

127. In most languages, a function definition like f a b = a : (f (a + b) b) would result in an infinite recursion. However, in Haskell we can partially evaluate functions like this because Haskell is based on ANSWER.
ANSWER=

128. Using the straightfoward expression translation scheme in the textbook, if I were to TransExp('3 * x + 1', vtable, ftable), newvar() will be invoked ANSWER times.
ANSWER=

129. In Prolog, the most natural way to express the fact that 'a lion is a cat' is ANSWER.
ANSWER=

130. In Haskell, instead of writing something like if x == 0 then 1 else fact (x - 1) * x, you can write a series of lines starting with | x ¿ 1 = x * factorial (x - a). This second style is called ANSWER.
ANSWER=

131. In Haskell, ANSWER is the output of zip [20..17][10, 8..4].
ANSWER=

132. In the context-free grammar $A \rightarrow BA$, $B \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$, $B \rightarrow$ the value of FOLLOW(A) is ANSWER.
ANSWER=

133. In the context-free grammar $A \rightarrow BA$, $B \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$, $B \rightarrow$ the value of FIRST(A) is ANSWER.
ANSWER=

134. In Prolog, the expression $[1, 2, 3] = [X|Y]$ causes X to have the value ANSWER.
ANSWER=

135. The routine in a compiler that takes as input a sequence of characters outputs these characters grouped into meaningful units is called ANSWER.
ANSWER=

136. Ruby supports two common ways that boolean expressions are handled in programming languages. In one approach both subexpressions of a boolean operator are evaluated before the boolean operator is evaluated. In the other approach, called ANSWER, the first subexpression in a boolean expression is evaluated and if that is enough to know the result of the boolean expression, then the second subexpression is not evaluated.
ANSWER=

137. Programming languages that view programming as describing characteristics of the problem domain and characteristics of the solution and leaving it to the language processor to find a solution are called ANSWER languages.
ANSWER=

138. In Haskell, instead of writing something like if x == 0 then 1 else fact (x - 1) * x, you can write a series of lines starting with factorial 0 = 1. This second style is called ANSWER.
ANSWER=

139. In Haskell, you use the keyword ANSWER to collect related code into a similar scope.
ANSWER=

140. The type of a string constant in Haskell, by default, is written ANSWER.
ANSWER=

141. In Prolog, the expression that would cause an unbound variable X to take on the sum of the values of a bound variable Y and a bound variable Z is ANSWER.
ANSWER=

142. The context-free grammar $A \rightarrow BA$, $B \rightarrow AB$, $A \rightarrow a$, $B \rightarrow b$, $B \rightarrow$ is not LL(1) specifically because ANSWER.
ANSWER=

143. The ! in Scala is used to ANSWER.
ANSWER=

144. In Ruby, the mixin is used to solve the object-oriented programming problem of ANSWER.
ANSWER=

145. When a function is invoked, if the language passes a copy of the value of each parameter to the code that performs the function, this is called ANSWER.
ANSWER=

146. A code block in Ruby is some lines of code surrounded by either curly braces or ANSWER.
ANSWER=

147. In the context-free grammar $A \rightarrow BA$, $B \rightarrow AB$, $A \rightarrow B$, $A \rightarrow a$, $B \rightarrow b$, and $B \rightarrow$ the value of FOLLOW(A) is ANSWER.
ANSWER=

148. Another design goal for Scala was to have its programs easily interoperate with those written in ANSWER.
ANSWER=

149. The Greek letter epsilon, when talking about languages, is used to represent ANSWER.
ANSWER=

150. One way of checking types is to see what constructor was used to create an object that is a parameter. Another way of checking types is to wait until a method is sent to an object and see if it supports the method. This second way is called ANSWER
ANSWER=

151. In Prolog, the expression hi(X, 4) = hi(3, X) causes X to have the value ANSWER.
ANSWER=

152. The idea that when a process has an error, it is up to a monitoring process to determine what to do about the problem is referred to by the motto ANSWER in Erlang.
ANSWER=

153. In Prolog, the expression X = 1 + 2 causes X to have the value ANSWER.
ANSWER=

154. The main programming paradigm for Clojure is ANSWER programming.
ANSWER=

155. In Ruby, the evaluation of arguments to a message are handled by the object sending the message. In Haskell, the runtime environment decides when and how much to evaluate an argument to a function. In Io, the evaluation of the arguments to a message is made by ANSWER.
ANSWER=

156. The technical term for connecting a name with an object is ANSWER.
ANSWER=

157. In the Erlang community, ANSWER code refers to replacing pieces of your application without stopping your application.
ANSWER=

158. In Ruby, if a line starts with a method name, that method is being sent to the object named ANSWER.
ANSWER=

159. Using the straightfoward statement translation scheme in the textbook, if I were to TransStat('if true then z := 1 else z := 2', vtable, ftable), newlabel() be invoked ANSWER times.
ANSWER=

160. In Scala, to indicate that a variable is immutable, you introduce it with the ANSWER keyword.
ANSWER=

161. In Haskell, how would you write an anonymous function so that the expression 'map ANSWER $[1, 2, 3]$' produces $[-4, -5, -6]$.
ANSWER=

162. Three concepts related to concurrency were discussed with regards to the language Io. ANSWER was presented as a way to request that something be computed and then be able to continue computing until the result was needed. If the result was available then things would proceed as expected. If the result was not available, then a wait would be initiated until the result became available.
ANSWER=

163. ANSWER typing is when the language implementation ensures that the arguments of an operation are of the type the operation is defined for.
ANSWER=

164. The main concurrency method used in Scala is ANSWER.
ANSWER=

165. When you send a message to a Ruby object, Ruby first looks at the methods that object supports, and then starts working the inheritance chain. If it still can't find the appropriate method, the message and its parameters get passed as a message to the object looking for a method called ANSWER.
ANSWER=

166. Scala is ANSWER typed
ANSWER=

167. In Ruby, the @ is used to indicate that the variable @me is ANSWER.
ANSWER=

168. In Haskell, if I define a function double = x + x, its type signature would be ANSWER.
ANSWER=

169. A programming language is called ANSWER if it is executed directly by an interpreter rather than by first being compiled with a compiler.
ANSWER=

170. In Haskell, if I write (h:t) = [3, 5, 7], ANSWER is the value of t.
ANSWER=

171. Each named object will have ANSWER, where the name is defined as a synonym for the object.
ANSWER=

172. In automatically generating the code that reads characters and outputs the part of a programming language that is analogous to its words, we start with a specification and then traditionally convert it into code in two stages. The main problem that can arise in moving from the first stage to the second stage is ANSWER.
ANSWER=

173. In Prolog, the expression hi(X, 4) = hi(3, Y) causes Y to have the value ANSWER.
ANSWER=

```
exam_database_file= examdatabase.json
exam_format= latex
dump_database= false
line_width= 72
question_count= 173
create_exam= false
answer_key= true
sample_seed= 222
shuffle_seed= 2345
["ICD1", "ICD2", "ICD3", "ICD4", "ICD5", "ICD6", "ICD9", "SLSW2",
"SLSW3", "SLSW4", "SLSW5", "SLSW6", "SLSW7", "SLSW8"]
["ICD1", "ICD2", "ICD3", "ICD4", "ICD5", "ICD6", "ICD9", "SLSW2",
"SLSW3", "SLSW4", "SLSW5", "SLSW6", "SLSW7", "SLSW8"]
```

1. Any
2. class
3. domain specific language
4. $E \Rightarrow (E) \Rightarrow ((E)) \Rightarrow ((1))$
5. boolean
6. - getname(id) was not declared
   - getname(id) was not bound
7. coroutines
8. LL(1)
9. - JVM
   - Java Virtual Machine
10. Actors
11. could be either true or false
12. open classes
13. a, b
14. dynamically typed
15. - a symbol table
    - an environment
16. Object
17. imperative
18. 5
19. $\leftarrow$
20. metaprogramming
21. tail recursion optimization
22. $[6, 8, 10]$

23. $(Numa) => a \to a \to a$
24. - >>=
    - a bind function
25. an activation record
26. strongly
27. an exit signal
28. forward
29. ++
30. dynamic typing
31. - cat(What).
    - cats(What).
    - is_a(What, cat).
    - are(What, cats).
32. type inferencing
33. a transaction
34. $[2, 3]$
35. - Domain Specific Languages
    - DSLs
36. list comprehensions
37. id indicates a token with a type and value field
38. ambiguous
39. clone
40. dangling else
41. trait
42. will always be true
43. [(17,10),(18,8),(19,6),(20,4)]
44. syntactic sugar
45. $[[1, 2], 3, 4]$
46. a recursive descent parser
47. - polymorphic
    - generic
48. Overloading
49. regular expressions
50. ($x \to 3 + x$)

51. the programmers
52. a start symbol
53. - a nondeterministic finite automata
    - a nondeterministic finite state machine
54. hash tables
55. XML
56. Immutability
57. call-by-reference
58. ancestor(I, You) :- parent(I, You).
59. a future
60. interface
61. 3
62. irb
63. static typing
64. Maybe
65. the address of the start of the function code
66. clone
67. return
68. 3
69. true
70. syntax-directed translation
71. $A \Rightarrow Ab \Rightarrow Abb \Rightarrow bbb$
72. call
73. A symbol table
74. - has a name that starts with a capital letter
    - has a slot for the method type
75. attributes
76. just-in-time compilation
77. second = head . tail
78. an interrupt
79. yield
80. $a(a|b)*$
81. type inference

82. - a sequence of terminals and nonterminals
    - a sequence of symbols
83. Nothing
84. - Software Transactional Memory
    - STM
85. a module
86. a symbol
87. - static scoping
    - lexical scoping
88. actors
89. - an infinite sequence of 1s
    - a lazy infinite sequence of 1s
90. - imperative
    - destructively updated
    - mutable
91. Class
92. inheritance
93. polymorphism
94. TrueClass
95. - inserting the association of getname(id) with the value v1 into the table
    - inserting the binding of getname(id) with the value v1 into the table
96. the argument list
97. self.
98. object-oriented and functional
99. currying
100. a class variable
101. - Rails
     - Ruby on Rails
102. - a finite automata
     - a finite state machine
103. Prolog
104. $X \Rightarrow XcY \Rightarrow XcYcY \Rightarrow cYcY \Rightarrow ccY \Rightarrow ccandX \Rightarrow XcY \Rightarrow XcX \Rightarrow XcXcY \Rightarrow cXcY \Rightarrow ccY \Rightarrow cc$
105. - persistent
     - functional
     - immutable

106. override
107. scope
108. strings
109. expressions that contain function usages
110. initialize
111. (4 8 16)
112. var
113. - X remains unbound
    - X remains unbound and the expression fails
114. - data and behavior are packaged together
    - there is a mechanism for restricting access to an object's components
115. where
116. - !
    - octal 41
    - decimal 33
    - hexidecimal 21
117. ancestor(I, You) :- parent(I, Ancestor), ancestor(Ancestor, You).
118. 3
119. threads
120. $E \to E + F$ and $E \to F$ and $F \to id$
121. false
122. a prototype
123. functional
124. tokens
125. expressions that contain identifiers
126. a code block
127. lazy evaluation
128. 5
129. - cat(lion).
    - is_a(lion, cat).
130. using guards
131. []
132. b
133. a, b

134. 1

135. - a lexical analyzer
     - a scanner
     - a lexer

136. short-circuit evaluation

137. declarative

138. pattern matching

139. module

140. [Char]

141. X is Y + Z

142. FIRST(BA) and FIRST(a) both include a, so we do not know which A rule to use

143. send a message to an actor

144. multiple inheritance

145. call-by-value

146. do end

147. a, b

148. Java

149. the empty string

150. duck typing

151. - X will not be bound and the expression will fail
     - X will not be bound

152. Let It Crash

153. 1+2

154. functional

155. the reciever of the message

156. binding

157. hot-swapping

158. self

159. 3

160. val

161. ($x \to -(x + 3)$)

162. Futures

163. Strong

164. actors
165. method_missing
166. statically
167. an instance variable
168. $(Numa) \Rightarrow a \rightarrow a$
169. interpreted
170. [5, 7]
171. a declaration
172. an exponential explosion in the number of states needed
173. 4