FINAL EXAM: CS3342b Tuesday, 25 April 2017, 2pm, Room FEB GYM

NAME AS APPEARS ON STUDENT ID:

STUDENT ID NUMBER:

GAUL/CONFLUENCE USER NAME:

REMINDERS:

1. (from course outline) 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.
2. Write neatly. If the marker can't read it, it is wrong.
3. This exam shouldn't take long to write. On the other hand, time will pass. It is a 3 hour exam with 50 questions. If you complete a question every 3 minutes (or 10 questions every half hour), you will still have a half hour at the end to double check that everything is in order.
4. While you are not allowed to open the exam booklet until the proctor says you can, you can fill out the information on the cover page. You should also get out your student id and make sure your pencils and pens are in order. If you need to get something out of your jacket or knapsack once the exam has started, raise your hand and wait til a proctor comes to you to oversee the matter.

1. 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=

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

3. 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() will be invoked ANSWER times.
   ANSWER=

4. 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=

5. 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=

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

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

8. 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=

9. 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=

10. 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=

11. 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=

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

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

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

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

16. 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=

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

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

19. 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=

20. 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=

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

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

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

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

25. 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=

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

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

29. 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=

30. 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=

31. 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=

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

33. 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=

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

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

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

37. 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=

38. 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=

39. 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=

40. 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=

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

42. 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=

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

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

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

46. 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=

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

48. 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=

49. 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=

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

```
exam_database_file= examdatabase.json
exam_format= latex
dump_database= false
line_width= 72
question_count= 50
create_exam= false
answer_key= true
sample_seed= 2322
shuffle_seed= 245
["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. the reciever of the message
2. - an infinite sequence of 1s
   - a lazy infinite sequence of 1s
3. 3
4. - 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
5. second = head . tail
6. multiple inheritance
7. - X will not be bound and the expression will fail
   - X will not be bound
8. - call-by-value
   - pass-by-value
9. ancestor(I, You) :- parent(I, Ancestor), ancestor(Ancestor, You).
10. the programmers
11. id indicates a token with a type and value field
12. boolean
13. an instance variable
14. ←
15. - static scoping
    - lexical scoping
16. true
17. syntax-directed translation
18. dynamic typing
19. - call-by-reference
    - pass-by-reference

20. $\{a, b\}$
21. A symbol table
22. - tail recursion optimization
    - tail recursion elimination
23. where
24. Any
25. - a sequence of terminals and nonterminals
    - a sequence of symbols
26. strongly
27. hot-swapping
28. an exit signal
29. using guards
30. lazy evaluation
31. return
32. currying
33. Actors
34. a prototype
35. domain specific language
36. Strong
37. just-in-time compilation
38. Immutability
39. hash tables
40. FIRST(BA) and FIRST(a) both include a, so we do not know which A rule to use
41. - a finite automata
    - a finite state machine
42. Maybe
43. clone
44. a declaration
45. Java
46. an exponential explosion in the number of states needed
47. - persistent
    - functional
    - immutable

48. 5

49. - JVM
    - Java Virtual Machine

50. type inferencing