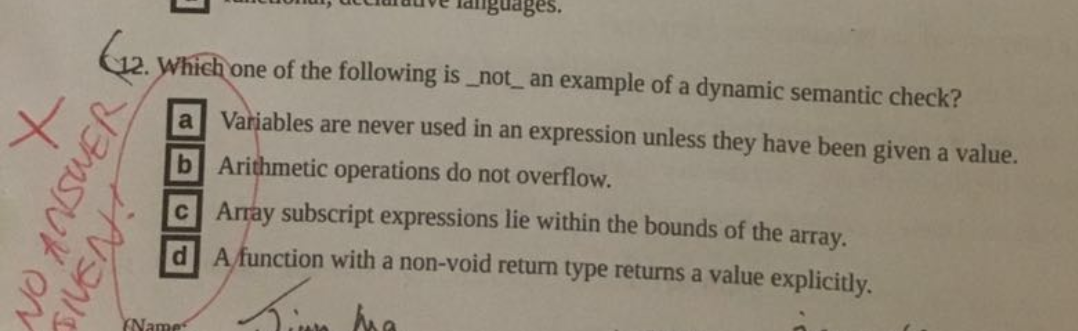
1. What ware the first high level programming language?
   1. Python
   2. C
   3. Ada
   4. FORTRAN
2. C, Ada, and FORTRAN are example of \_\_\_\_\_\_\_\_ Programming language
   1. Von Nenmann
   2. Functional
   3. Constraint\_based
   4. Logic
3. All of the following are phases of a compiler, excepts
   1. Syntactic Analysis
   2. Numerical Analysis
   3. Lexical Analysis
   4. Semantic Analysis
4. In the Dataflow model of computation, a program is
   1. Ultimately based on the modification of variables
   2. A function written in terms of simpler function, eventually based on primitives.
   3. A set of specified relationships solved via goal-directed search through logical rules.
   4. A graph of functional nodes, each triggered by the arrival of inputs and producing output.
5. The distinctions between language categories, families, etc, are precise and clear-cut.
   1. False
   2. True
6. Source-to-source compiling is when a compiler….
   1. Translates a source program to assembly language instead of machine code.
   2. Translates a source program in one high-level language to another high-level language
   3. Performs only a partial analysis of the source program
   4. Performs only textual processing of the source program
7. Good reason to study programming languages include..
   1. Making better use of a language you already “know”.
   2. Making it easier to learn a new language.
   3. To satisfy some arbitrary and bogus requirement for a degree.
   4. A and b
8. Which\_one\_of the following best describes P-code?
   1. It is machine language for the P86 target processor.
   2. It is a high-level programming language
   3. It is code that is specific to a target processor.
   4. It is an intermediate code.
9. It is possible for a language to have both declarative\_and\_imperative characteristics.
   1. True
   2. False
10. An interpreter
    1. Is more efficient than a compiler.
    2. Translates the source program every time it is to run.
    3. Manipulates the source program and then goes away.
    4. Converts the source program into a target program in machine code.
    5. A and b
11. In the object oriented model of computation, a program is…
    1. A graph of functional nodes, each triggered by the arrival of inputs and producing output.
    2. A function written in terms of simpler functions, eventually based on primitives.
    3. An interaction among (semi-) independent entities, each with its own state and routines to modify that state.
    4. A set of specified relationships solved via goal-directed search through logical rules.
12. The best programming language for any applications is
    1. The one you boss tells you to use.
    2. The one you might learn something from.
    3. The one you colleagues are using.
    4. A, b, or C
13. The back end of a compiler is concerned with?
    1. Generating appropriate target code.
    2. Determining what a program means
    3. Performing dynamic semantic analysis
    4. Both lexical and syntactic analysis
14. Syntactical analysis converts \_\_\_\_\_ into \_\_\_\_\_
    1. Tokens //// a parse tree
    2. Parse trees //// intermediate code
    3. Heap entries //// stack frames
    4. Symbol tables //// lexical units
15. Compilation and interpretation are so different they cannot be used together
    1. True
    2. False
16. The hardware used to construct essentially all computers is \_\_\_\_\_ in nature
    1. Declarative
    2. Non-deterministic
    3. Imperative
    4. Non-euclidean
17. Semantic Analysis discovers a program’s
    1. Runtime complexity.
    2. Syntactic correctness
    3. Meaning
    4. Space requirements
18. Is it generally possible to compile a program written in an interpreted language?
    1. Yes, but only if the program is in the functional or dataflow subset of the interpreted language
    2. No, never. That is why the language was interpreted in the first place.
    3. Yes, but it is not worth doing so because of the great decrease in performance
    4. Yes, when assumptions are made and there is a fall-back process to handle violations
19. Declarative programming languages…
    1. Are significantly higher-performing than imperative languages.
    2. Focus on\_what\_the program is to do.
    3. Try to get away from “irrelevant” implementation details
    4. A and b
    5. B and c
20. In JIT, compilation is…
    1. Performed as soon as possible.
    2. Delayed until the last possible moment.
    3. Repeated until the code is of minimal size.
    4. Eliminated in favor of interpretation.
21. Pretty much all “traditional” languages (Ada, Algol, BASIC, C, COBOL, FORTRAN, Pascal, ...) are …
    1. Von Neumann-style, imperative languages.
    2. Script-based, imperative languages.
    3. Dataflow, declarative languages.
    4. Functional, declarative languages.
22. Which one of the following is\_not\_an example of a dynamic semantic check?
    1. Variables are never used in an expression unless they have been given a value.
    2. Arithmetic operations do not overflow.
    3. Array subscript expressions lie within the bounds of the array.
    4. A function with a non-void return type returns a values explicitly
    5. 
23. In general, all programming languages have the same computation power.
    1. False
    2. True
24. LISP, Scheme, ML. and Haskell are examples of \_\_\_\_\_ programming language
    1. Functional
    2. Scripting
    3. Constraint-based
    4. Von Neumann
25. Lexical analysis converts \_\_\_\_ into \_\_\_\_
    1. Parse trees ///// symbol tables
    2. Integers ///// floating pint numbers
    3. Semantics ///// syntax
    4. Characters ///// tokens
26. Saying that a programming language is Turing complete mean …
    1. It is completely imperative in nature
    2. It has the same computation power as a Turing Machine
    3. It was completed by Alan Turing.
    4. It is completely declarative in nature.
27. The front end of a compiler is concerned with ….
    1. Machine-specific code optimization
    2. Determining what a program means.
    3. Performing dynamic semantic analysis
    4. Generating appropriate target code
28. Which one of the following is\_not\_an example of a dynamic semantic check?
    1. Variables are never used in an expression unless they have been given a value
    2. Arithmetic operations do not overflow
    3. A function with a non-void return type return a value explicitly
    4. Array subscript expression lie within the bounds of the array.
29. Imperative programming languages …
    1. Though lower-performing than declarative languages, are more expressive
    2. Focus on\_how\_the program is to accomplish its function
    3. Align better with the structure of the underlying hardware.
    4. A and b
    5. B and c
30. C, Ada and Fortran are examples of \_\_\_\_ programming languages.
    1. Constraint-based
    2. Logic
    3. Functional
    4. Von Neumann
31. In the von Neumann model of computation, a program is
    1. An interaction among (semi-)independent entities, each with its own state and routines to modify that state
    2. A function written in terms of simpler functions, eventually based on primitives
    3. Ultimately based on the modification of variables
    4. A set of specified relationships solved via goal directed search through logical rules
32. The first assembler was written
    1. An the University of Texas / Arlington
    2. Using p-code as an intermediate representation
    3. By means of non-deterministic finite automata
    4. In machine code directly.
33. The initial goals for LISP included
    1. Being a practical mathematical notation for computer programs
    2. Communicating numerical procedures between people
    3. Supporting the investigation of artificial intelligence
    4. A and c
    5. B and c
34. The von Neumann, object-oriented, and scripting language families are distinct and do not overlap
    1. True
    2. False
35. Which one of the following is\_not\_an\_example of a static semantic check
    1. Every identifier is declared before it is used
    2. Labels on the arms of a switch statement are distinct constants
    3. Subroutine calls provide the correct number and types of arguments
    4. Pointers are never dereferenced unless they refer to a valid object
36. Static semantic check
    1. Depend on both the structure and the execution of the program and cannot be enforced until runtime
    2. Depend solely on the structure of the program and can be enforced at compile time
    3. May depend on the execution of the program and must be enforced at runtime
    4. Have nothing to do with structure or execution; they are purely concerned with syntax
37. The initial goals for ALGOL included
    1. Communicating numerical procedures between people
    2. Realizing procedures on a variety of machines
    3. Supporting cross-border financial transactions
    4. A and b
38. An interpreter and a compiler are similar in structure
    1. Only in the target machine code generation phase
    2. Not at all. They are completely different in structure
    3. Through the front end.
    4. Through the back end
39. Creating the first FORTRAN compiler took about \_\_\_\_ because \_\_\_\_\_
    1. Only a few months ///// compiling has been so intently studied
    2. 18 staff years ///// it was the first compiler written
    3. 18 staff months ///// it was written an Algol
    4. 18 staff years ///// they already had algol and cobol as example
40. Why are compilers able to detect more static semantic errors than inter
    1. Compilers generate target machine code which detects these kinds of errors.
    2. Compilers use tree-walk routines which are useful for detecting these kinds of problems
    3. Compilers perform semantic analysis and interpreters do not
    4. Compilers examine the entire code structure.
41. Declarative languages generally include some imperative constructs
    1. False
    2. True
42. The initial goals for COBOL included
    1. Supporting the investigation of artificial intelligence
    2. Being a portable data processing programming language
    3. Being self-documenting and readable by non-programmers
    4. A and C
    5. B and C
43. The first compiler was written
    1. In secret during world war 1
    2. Under contract to the us department of defense
    3. In a mixture of assembler and machine code
    4. In FORTRAN
44. a compiler
    1. processes the source program every time it needs to be executed
    2. is more flexible and has better diagnostics than an interpreter
    3. translates a source program into a target program and then goes away
    4. is a “virtual machine” whose machine language is the programming language
    5. b and C
45. which\_one\_of following does a preprocessor do?
    1. It performs macro expansions
    2. It makes code machine independent
    3. It performs syntax analysis of the code
    4. It generates output in assembly language