1. An optimization is said to be \_\_\_\_ if \_\_\_\_
   1. Optimistic //// it always decreases memory usage.
   2. Speculative //// it usually improves performance but may degrade in some cases
   3. Conservative //// it never generates a memory fault.
   4. A b and C
2. Assertions are ….
   1. Used to check the correctness of a program
   2. Expected to be true when execution reaches them
   3. Created by the compiler
   4. A and b
   5. A and C
3. Syntax concerns the \_\_\_\_\_ of a program, while semantics concerns its \_\_\_\_\_
   1. Completeness //// efficiency
   2. Meaning //// from
   3. Form //// meaning
   4. Efficiency ///// completeness
4. Semantic analysis is required for
   1. Determining information required for the generation of the output program
   2. Enforcement of the rules for a valid program that go beyond the form of the program
   3. Ensuring invalid programs do not violate dynamic checks
   4. A and b
   5. A and c
5. C has what level of dynamic semantic checking ?
   1. None, C does no dynamic semantic checking
   2. Some
   3. High
   4. Medium
6. A postcondition failure indicates a problem with \_\_\_\_\_
   1. A caller
   2. a Callee
   3. The encompassed code
   4. None of the above
7. Resolution of overloaded names is carried out by the
   1. Semantic analyzer
   2. Syntactic analyzer
   3. Intermediate code generator
   4. Regular expression processor
8. A precondition failure indicates a problem with \_\_\_\_
   1. The encompassed code
   2. A callee
   3. A caller
   4. None of the above
9. Static semantic rules are those that can be enforce at
   1. Link time
   2. Run time
   3. Program writing time
   4. Compile time
10. Optimistic compilers insert Dynamic checks to undo the effects of unsafe code when it goes wrong
    1. False
    2. True
11. Alias Analysis can be used to determine when it is safe to
    1. Allow access by concurrent threads
    2. Evaluate expressions out of order
    3. Cache values in registers
    4. A and b
    5. A b and c
12. Dynamic type checking can never be completely safe
    1. False
    2. True
13. A compiler is optimistic if it applies only optimizations that are both safe and effective
    1. False
    2. True
14. A static analyzer is said to be precise if it allows the compiler to determine whether a program will always follow a rule
    1. False
    2. Ture
15. A compiler is optimistic if it applies speculative or unsafe optimizations
    1. True
    2. False
16. Java’s dynamic semantic checking helps prevent inadvertent errors
    1. Yes that was a prime goal in the design of the language
    2. No java has no dynamic semantic checking at all
    3. No, dynamic semantic checking has nothing to do with detecting run time errors
    4. No this capability is intended to improve heap management
17. Comparing argument lists in function calls to that given in the function’s declaration is an example of a
    1. Semantic check
    2. Chomsky hierarchy
    3. Syntactic check
    4. None of the above
18. AN optimization is said to be \_\_\_\_\_ if \_\_\_\_\_
    1. Conservative ////// it never generates a memory fault
    2. Unsafe ////// it might lead to incorrect code in a program
    3. Optimistic //// it always decreases memory usage
    4. A, b and c
19. Invariants, preconditions, and postconditions are statements about the \_\_\_\_ of the program
    1. Size
    2. Efficiency
    3. Correctness
    4. Speed
    5. None of the above
20. Language \_\_\_ the kinds of semantic rules they enforce
    1. Tend to change
    2. Differ greatly in
    3. Are the same in
    4. A and c
21. Syntax concerns the \_\_\_\_ of a program, while semantics concerns its \_\_\_\_\_
    1. Meaning //// form
    2. Efficiency ///// completeness
    3. Form //// meaning
    4. Completeness /// efficiency
22. Static semantic rules are those that can be enforced at
    1. Compile time
    2. Program writing time
    3. Link time
    4. Run time
23. Ada assigns a \_\_\_\_ to every numeric variable and \_\_\_\_
    1. Heap location //// compacts whenever fragmentation occurs
    2. Specific type //// requires the user to make explicit conversion
    3. Dynamic allocation strategy ///// converts the base as required
    4. Generic type ///// performs any required conversions
24. Languages\_\_\_\_\_ the kinds of semantic rules they enforce
    1. Differ greatly in
    2. Tend to change
    3. Are the same in
    4. B and C
25. A compiler is conservative if it applies only optimizations that are both safe and effective
    1. True
    2. False
26. A static semantic rules are those that can be enforced at
    1. Link time
    2. Program writing time
    3. Run time
    4. Compile time
27. Assertions are \_\_\_\_ regarding \_\_\_\_\_
    1. Static checks //// the parse Tree.
    2. Processed at link time //// the Fragmentation of Heap
    3. Logical formulae ///// the values of program data
    4. Created by the compiler ///// the lexical structure of the program
28. Semantic analysis includes items that are not convenient to capture in the language’s
    1. Regular expressions
    2. Context free grammar
    3. Chomsky hierarchy
    4. A and C
29. Generally, the semantic analyzer produces the \_\_\_\_\_ used by the \_\_\_\_\_
    1. Central reference table ////// Association
    2. Heap linked list ///// regular expression parser
    3. Memory management index //// lexical analysis phase of compilation
    4. Annotated syntax tree//// code generation phase of compilation
30. an invariant failure indicates a problem with \_\_\_\_
    1. a Caller
    2. the encompassed code
    3. a callee
    4. none of the above
31. Semantic analysis Can require the presence of a return statement in a function easily
    1. True
    2. False
32. Conservative compilers insert static checks to undo the effects of unsafe code when it goes wrong
    1. False
    2. True
33. Identifying the use of an undeclared variable is an example of a \_\_\_\_ check
    1. Static semantic
    2. Dynamic semantic
    3. Dynamic syntactic
    4. Static syntactic
34. Compile-time algorithms that predict run-time behavior are known as
    1. Static analyzers
    2. Dynamic analyzers
    3. A or b
    4. None of the above
35. Roles of the semantic analyzer include
    1. Annotating the parse tree with any information needed by the intermediate code generator
    2. Generating machine code for all relevant symbol table entries
    3. Compacting the storage required for any information needed by the central reference table
    4. Relocating the heap index table for the lexical analyzer
36. Java has what level of dynamic semantic checking ?
    1. Medium
    2. Some
    3. High
    4. Non, java does no dynamic semantic checking.
37. Subtype analysis helps in determining whether heap allocation can be used instead of stack allocation
    1. False
    2. True
38. Loop invariants are normally checked
    1. When the enclosing function is dynamically loaded
    2. Before and after each loop iteration
    3. Only at compile time for increased efficiency
    4. During subroutine entry and exit
39. Identifying an array index out of bounds error is generally an example of a \_\_\_\_\_\_check
    1. Static syntactic
    2. Static semantic
    3. Dynamic semantic
    4. Dynamic syntactic
40. Semantic analysis includes items that are not convenient to capture in the language’s
    1. Context free grammar
    2. Regular expressions
    3. Chomsky hierarchy
    4. B and C
41. \_\_\_\_ are expected to be true at all “clean points” of a given body of code
    1. Postconditions
    2. Preconditions
    3. Invariants
    4. Non of the above
42. \_\_\_\_\_ are expected to be true at the beginning and end of subroutines
    1. Preconditions
    2. Postconditions
    3. A and B
    4. Non of the above
43. “Design by Contract” interface specification are
    1. Syntactic, semantic, and dynamic
    2. Informal, efficient, and static
    3. Lexical, semantic and of minimal size
    4. Formal, precise, and verifiable
44. In C and C++, asserts
    1. Are implemented with a macro
    2. Are usually enabled in production runs
    3. Are usually disabled after debugging
    4. A and b
    5. A and c
45. Static and precise type checking ensures that a variable will always be used in a type-safe manner
    1. False
    2. True
46. Optimistic compilers insert static checks to undo the effects of unsafe code when it goes wrong
    1. False
    2. True
47. An optimization is said to be \_\_\_\_\_ if \_\_\_\_\_
    1. Conservative //// it never generates a memory fault
    2. Unsafe //// it might lead to incorrect code in a program
    3. Optimistic //// it always decreases memory usage
    4. A b, and C
48. Escape analysis helps in determining whether dynamic method dispatch can be avoided
    1. True
    2. False
49. Roles of semantic analyzer include
    1. Enforcing all syntactic rules
    2. Enforcing all parsing rules
    3. Enforcing all static rules
    4. Enforcing all lexical rules
50. Python provides \_\_\_\_\_ integer operations
    1. Syntactically rounded
    2. Statically allocated
    3. Essentially infinite precision
    4. Finite precision
51. Java performs all type checking at compile time
    1. True
    2. False
52. Context free grammars can require the presence of a return statement in a function easily
    1. True
    2. False
53. Generally, any rule requiring comparing separated items or counting items will be a \_\_\_\_check rather than a \_\_\_\_ check
    1. Dynamic //// static
    2. Semantic //// syntactic
    3. Syntactic //// static
    4. Semantic ///// dynamic
54. C’s dynamic semantic checking helps catch malicious errors
    1. No, C has no dynamic semantic checking at all
    2. Yes, and it also improves heap management
    3. Yes, that was a prime goal in the design of the language
    4. No, this capability is intended to improve heap management
55. Syntax concerns the \_\_\_\_ of a program, while semantics concerns its -\_\_\_\_
    1. Meaning /// form
    2. Form //// meaning
    3. Efficiency ///// completeness
    4. completeness ///efficiency
56. semantic rules may be either \_\_\_\_ or \_\_\_\_\_
    1. theoretic /// pragmatic
    2. syntactic /// lexical
    3. static /// dynamic
    4. formal /// ad hoc
57. The semantic analyzer annotates the \_\_\_\_with any information needed to \_\_\_\_
    1. Lexical structure //// parse the regular expressions
    2. Memory hierarchy //// compact the heap when it’s fragmented
    3. Symbol table /// execute more efficient code
    4. Parse tree //// generate dynamic semantic checks