1. Compilers may apply mathematical and logical identities to expression evaluation
   1. Always, even overriding parentheses, if the code can be made more efficient
   2. Only when specifically indicated by the user
   3. Unless forced not to by parentheses
   4. Never
2. In recursion, a section of code is
   1. Defined in terms of selection and iteration only
   2. Defined in terms of regular expression
   3. Defined in terms of itself, either directly or indirectly.
   4. Defined in terms of expression-level abstraction.
3. Changing the value of the loop iteration variable inside the loop results in
   1. Undefined behavior
   2. A decrease in stack space usage usually
   3. An increase in execution speed, usually
   4. None of the above
4. A compound statement is
   1. A recursive technique that improves speed in an imperative language
   2. A matching of two statements to increase efficiency
   3. A concurrent set of expressions that acts as a statement
   4. A grouping of statements that acts as a single statement
5. Tail recursion can be converted into
   1. Iteration
   2. Selection
   3. Non-determinacy
   4. Sequencing
6. In java primitive type are \_\_\_\_\_\_ and object types are \_\_\_\_\_
   1. Compile time constant ///// run time evaluated
   2. Values ///// references
   3. Heap based ////// register based
   4. preallocated //// postatllocated
7. sequencing is central concept in
   1. imperative languages
   2. functional languages
   3. dataflow languages
   4. logical languages
8. side effects are fundamental to
   1. imperative languages
   2. dataflow languages
   3. logic languages
   4. functional languages
9. FORTRAN’s assigned and computed goto statements
   1. Have been copied by most modern languages because of their efficiency
   2. Are still emphasized in modern usage because there is no better way to write code
   3. Were more useful early on because FORTRAN had no better constructs
   4. A and b
10. Recursion is a central concept in
    1. Von Neumann languages
    2. Functional languages
    3. Imperative languages
    4. None of the above
11. The goto statement used to be considered bad, but now is
    1. Considered OK as long as each use is clearly documented
    2. Being retro-fitted to languages that originally had no goto
    3. Part of every modern language
    4. Still considered quite dangerous in general
12. The use of an uninitialized variable
    1. Doesn’t matter, as all languages define default variable values anyway
    2. Can never be detected that would require a solution to the halting problem
    3. Can sometimes be detected, depending on the expression
    4. Can always be detected, though it might be inefficient to do so.
13. Boxing and unboxing are used when \_\_\_\_ must be used together
    1. Global objects and local objects
    2. Static types and dynamic types
    3. Heap objects and stack objects
    4. Value types and reference types
14. Case or switch statements
    1. Are often clearer and cleaner than long if / then / elsif/ elsif/ …chains
    2. Are less efficient than long if / then / elsif / elsif/ …chains
    3. Can be implemented by the compiler in many ways
    4. A and C
    5. B and C
15. A counted style of loop has
    1. A stack value, a heap value, and a global value
    2. A precondition, a postcondition and an invariant
    3. A left value, a right value, and a local value.
    4. A starting value, an ending value, and a step value.
16. An iterator
    1. Enumerates the elements of a well-defined set
    2. Extends logically controlled loops to include non-determinacy
    3. Calculated loop iteration counts using floating point arithmetic instead of integers
    4. Extends iteration to include recursion
17. Definite assignment requires that every variable can be shown to have
    1. An assigned value on each path before its use
    2. A assigned value on more tha none path before its use
    3. An assigned value on at least one path before its use
    4. A non-zero value before its use
18. In logically-controlled loop the rest must be
    1. In the middle of the loop
    2. At the end of the loop
    3. At the beginning of the loop b or c
    4. A b or c
19. An assignment is generally executed
    1. To reduce the amount of heap space required for recursion
    2. Only at the beginning of the fetch-execute cycle, not the end
    3. Solely for its side effect, the change in the value of the 1-value
    4. In order to improve the efficiency of functional code
20. Even in strict imperative language, a compiler might reorder statements for optimization reasons
    1. False
    2. True
21. two forms of iteration
    1. enumeration controlled and logically controlled
    2. compile time and run time
    3. heap based and stack based
    4. static and dynamic
22. Various kinds of control flow include
    1. Nondeteminacy
    2. Speculation
    3. Exception handing
    4. A and b
    5. A b and c
23. In selection
    1. A choice is made of two or more statements or expression
    2. A recursion is made of a functional expression
    3. A repetition is made of at least two expressions
    4. A statement is executed depending on a non-deterministic condition
24. Any recursion can be rewritten as iteration
    1. False
    2. True
25. Most operators are
    1. Left-associative, binding from left-to right
    2. Both left-and right associative, depending on the expression
    3. Neither left-nor right- associative, as that is for machine languages only
    4. Right-associative, binding from right-to-left
26. The C for statement
    1. Allows any changes the user want to variables inside its body
    2. Allows updating the loop interation variable but only in the negative direction
    3. Allows updating the loop interation variable but only in the positive direction
    4. Prohibits the changing of the loop iteration variable inside its body
27. Assignment operators are generally
    1. Right-associative, binding from right-to-left
    2. Neither left-nor right associative as that is for machine languages only
    3. Left-associative, binding from left to right
    4. Both left-and right associative depending on the expression
28. The relationship between iteration and recursion is that
    1. Recursion can express algorithms that iteration cannot
    2. Though logically equivalent, recursion is always more efficient than iteration
    3. Iteration can express algorithms that recursion cannot
    4. They are logically equivalent. Using one or the other is personal preference.
29. All languages have
    1. Their own particular sets of operators
    2. The same operators since they are defined by the hardware
    3. The same operators since they are defined by industry standards
30. Recursion requires
    1. Dynamic semantic checks to ensure higher performance than iteration
    2. A clear indication of recursion steps via syntactic markers
    3. No special syntax in general
    4. Direct references as indirect ones would require a solution to the halting problem
31. An r-value indicates
    1. The size of the current stack frame allocation
    2. The argument list of a recursive function
    3. The expression to use in a loop termination check
    4. The new value to store in an assignment
32. When a loop exits the value of the loop index variables is
    1. The loop ending value plus step
    2. Undefined
    3. Zero
    4. [it depends on the language]
33. A side effect can cause
    1. Significant speed up in recursive code but has no effect on iterative code
    2. No bad effects because the compiler guarantees correct code always
    3. Iteration, selection, and sequencing to generate the same answers
    4. The evaluation of the same expression to possibly have a different result
34. Lisp uses
    1. Postfix polish notation
    2. Infix polish notation
    3. Reverse polish notation
    4. Cambridge polish notation
35. An l-value indicates
    1. The value to use a loop index calculation
    2. The size of a heap allocation request
    3. The address of a recursive function
    4. The location to change in an assignment
36. Multiway assignment allows
    1. Decreased memory use when multiplying integers
    2. Improvement in stack trace generation
    3. Increased efficiency in floating point operations
    4. Exchanging two variables without using a temporary
37. In an assignment, a variable name may appear
    1. As an r-vlaue
    2. As neither an l-nor r value
    3. As an l-value
    4. A or C
38. Structured programming emphasized
    1. Top down design
    2. Recursion in preference to iteration
    3. Code modularization
    4. A and b
    5. A and C
39. All languages have
    1. The same operators since they are defined by industry standards
    2. The same operators since they are defined by the hardware
    3. Their own particular sets of operators.
40. The order of operand and function call argument evaluation is
    1. Unspecified
    2. Right – to-left
    3. Either left-to-right or right to left, depending on the expression
    4. Left to right
41. (\*(+ 1 2 )3) is an example of
    1. Postfix polish notation
    2. Reverse polish notation
    3. Cambridge polish notation
    4. Infix polish notation
42. The two models of assignment are
    1. The value and the reference models
    2. The formal and the abstract models
    3. The von Nenmann and the Chomsky models
    4. The specific and the generic models
43. The c for statement is …
    1. An enumeration controlled loop statement
    2. Neither logically nor enumeration controlled
    3. Either logically-or enumeration controlled depending on the expressions
    4. A logically controlled loop statement
44. Operators can be
    1. Berkeley, Cambridge or Harvard style
    2. Infix, prefix, or postfix
    3. Forward, backward, or reversed formatted
    4. Left, middle, or right ordered
45. Operators are
    1. Are never user-defined
    2. Special, short names for certain functions
    3. Are always user-defined
    4. A and b
    5. B and C
46. Assignment have
    1. A function value and a subroutine value
    2. An l-value and r value
    3. A statement value and an expression value
    4. A prefix value and a postfix value
47. Orthogonality is when language features may be used
    1. At both compile-time and run-time
    2. Both recursively and iteratively
    3. In any combination with consistent meaning
    4. Statically and dynamically
48. Combination assignment operators are used
    1. To improve readability
    2. To increase safety
    3. To improve efficiency
    4. B and c
    5. A, b and c
49. Non-determinacy is when
    1. Speculation is shown to be both conservative and optimistic
    2. The order of execution or selection of alternative is left unspecified
    3. Statement execution happens with maximal concurrency
    4. Selection is based on integer operations rather than logical ones
50. Floating point limits for loops are OK because
    1. Floating point arithmetic is usually more precise than integer arithmetic
    2. Floating point arithmetic is usually more efficient than integer arithmetic
    3. Floating point arithmetic is defined by a standard
    4. No floating point arithmetic is not ok for loop limits
51. Ambiguity of pairing an else clause with its matching if is known as
    1. The multi-test determination problem
    2. The decision statement reordering problem
    3. The selection sequencing problem
    4. The dangling-else problem
52. Eliminating side effects can be helpful in
    1. Proving attributes about programs
    2. Improving program safety and correctness
    3. Making programs easier to understand
    4. Easing program optimization
    5. A b c and d
53. Almost any well-defined imperative algorithm can be expressed with only
    1. Selection, alteration and conditionals
    2. Sequencing, selection, and iteration
    3. Exception-handing, speculation, and procedural abstraction
    4. Recursion, iteration and non-determinacy
54. In iteration a section of code is
    1. Executed in the abstract, from a procedural point of view
    2. Executed once, in a non-deterministic fashion
    3. Executed in alternation with another section of code
    4. Executed repeatedly, usually for a given number of times or until a condition is satisfied
55. A function call may be executed for
    1. Its return value
    2. A side effect
    3. Iterative reasons only
    4. A or b
    5. A or c
56. In concurrency, multiple section of code
    1. Are executed at the same time
    2. Are recompiled to ensure only register-to-register transfers are used
    3. Are restated in terms of expression-level recursion
    4. Are forcibly distributed across a local area network
57. Short-circuit operators
    1. Act just like other operators, but are more efficient
    2. Have a defined evaluation order and may leave operands unevaluated
    3. Have no specified evaluation order and evaluate all operands
    4. Act just like other operators but take up less space
58. Floating point uninitialized variable use can be detected by
    1. Using infinity as the default value and then watching for an overflow
    2. Using Nan as the default value and then watching for a hardware exception
    3. Using negative zero as the default value and then watching for an incorrect result
    4. B or c
59. In logically controlled loops the test must be
    1. At the beginning of the loop
    2. At the end of the loop
    3. In the middle of the loop
    4. A or b
    5. A b or c
60. Exponentiation is generally
    1. Right-associative, binding from right-to-left
    2. Left-associative binding from left-to-right
    3. Both left-and right associative, depending on the expression
    4. Neither left-nor right associative as that is for machine languages only
61. Repeat / until and do /while are examples of
    1. Logically controlled loop styles
    2. Enumeration controlled loop styles
    3. Non-deterministic loop styles
    4. Recursion-based loop styles