1. Most languages have \_\_\_\_ modules
   1. Open scope
   2. Dynamic scope
   3. Closed scope
   4. Opaque scope
2. A forward reference is
   1. A Reference to an object that has not yet been allocated
   2. A reference to a function that has not yet been loaded
   3. A reference to a name that has not yet been declared.
   4. A reference to a program that has not yet been compiled.
3. Which of the following is are true about static scoping?
   1. Scope determination can be made by the compiler
   2. The bindings cannot be resolved by examining the program text
   3. Static scoping tis an improvement on lexical scoping
   4. In general compiled language do not use static scoping
   5. B and d
4. Most compiled languages use
   1. Dynamic scoping rules
   2. Lexical scoping rules
   3. A mixture of lexical and dynamic scoping rules
   4. Regular expression scoping rules
5. In java, can a name be used before it is declared?
   1. No, never
   2. Yes, for class methods or class properties
   3. Yes , for standard package names in the global namespace
   4. Yes, for local variables in a class methods
6. Names in a programming language are a way to…
   1. Refer to program elements symbolically.
   2. Increase run-time efficiency
   3. Reduce conceptual complexity by hiding irrelevant details
   4. Decrease the required memory size
   5. A and C
7. A function’s local variables must always be allocated on the stack
   1. Yes, because static allocation cannot be done for local variables
   2. Yes, because the compiler does not know how big they are.
   3. No, not necessarily depending on the language
   4. Yes, because formal methods have been used to prove this.
8. 8. Lexical scoping
   1. Requires Cohen-Sutherland symbol table.
   2. Is defined in terms of the textual layout of the program.
   3. Can be determined by the compiler.
   4. A and b
   5. B and c
9. Why are stacks used to hold local variables?
   1. To support recursion
   2. To support dynamic binding
   3. To make more efficient use of space
   4. A and b
   5. A and c
10. An object that outlives all of its reference is known as
    1. Garbage
    2. A dandling object
    3. A shadow object
    4. A dead object
11. Nested blocks
    1. Sometimes cause scope holes.
    2. Never cause scope holes
    3. Always cause scope holes
    4. Have noting to do with scope holes.
12. External fragmentation of a heap is when requests overflow the bounds of the stack
    1. True
    2. False
13. Modules facilitate \_\_\_\_\_ by \_\_\_\_
    1. Security ///// implementing a staged allocation strategy
    2. Abstraction ///// allowing data to be made private
    3. Formalization ///// requiring detailed specifications
    4. Efficiency ///// reducing overall stack and heap usage
14. with static scoping, a symbol table must be maintained at run time
    1. False
    2. True
15. Reference counting, Mark / Sweep, Copying, and Generational are kinds of
    1. Dynamic binding strategies
    2. Garbage collection
    3. Regular expression analysis
    4. Lexical parsing methods.
16. A name that can refer to two or more objects is said to be
    1. Overloaded
    2. Polymorphic
    3. Coerced
    4. Aliased
17. The referencing Environment is
    1. The extent of the lexical scope in the current function
    2. The collection of local variable in the current stack frame
    3. The set of active bindings
    4. The entire contents of the compiler’s symbol table
18. Commonly, unintended aliasing occurs because of
    1. Closed-scope modules
    2. Heap allocation
    3. Reference parameters
    4. Reference parameters
    5. Dynamic methods dispatch
    6. A and d
19. Which of the following is/are true about dynamic scoping
    1. In general, compiled languages do not use dynamic scoping.
    2. Dynamic scoping is a formalization of lexical scoping
    3. The binding cannot be resolved by examining the program text.
    4. A and C
    5. A, b and C
20. Heap storage is preferred to stack storage when
    1. The object will be statically bound
    2. The object may outlive the function that creates it.
    3. The size of the object may vary.
    4. A and C
    5. B and C
21. Names in a programming language are a way to
    1. Increase run-time efficiency.
    2. Refer to program elements symbolically.
    3. Reduce conceptual complexity by hiding irrelevant details
    4. Decrease the required memory size
    5. B and C
22. C and C++ differentiate declarations and definitions
    1. To support recursive and mutually-referential types.
    2. To improve the readability of the program.
    3. To make the compilation process faster and more efficient.
    4. To encourage structured program complexity control
    5. B and C
23. Dynamic method dispatch is when..
    1. A parent class overrides a methods in a child and the selection is made at compile time
    2. A parent class overrides a methods in a child and the selection is made at run time
    3. A child class overrides a methods in a parent and the selection is made at run time
    4. A child class overrides a methods in a parent and the selection is made at compile time.
24. Symbol table information may be made available at run time
    1. To aid in the formalization of execution time semantics
    2. To increase the efficiency to local variable allocation
    3. To support symbolic debugging
    4. To decrease the required amount of memory space
25. Bindings may be made at
    1. Runtime
    2. Load time
    3. Compile time
    4. A and c
    5. A b and c
26. What type or types of binding does c have
    1. Dynamic binding
    2. Stack binding
    3. Heap binding
    4. Static binding
    5. A b c and d
27. Lexical scoping
    1. Is defined in terms of the textual layout of the program
    2. Requires Cohen Sutherland symbol table
    3. Can be determined by the compiler
    4. A and b
    5. A and C
28. Single-subroutine abstraction is not that useful because
    1. The methods is not support by most languages
    2. There is no compiler support for cross-platform use
    3. The data can be used only by a single subroutine
    4. A and b
29. With dynamic scoping, a symbol table must be maintained at run time
    1. True
    2. False
30. In selectively open scope modules
    1. Names are automatically exported but require qualification
    2. Names are explicitly exported but may be overridden
    3. Names are automatically imported but require explicit allocation
    4. Names are explicitly imported but may be shadowed
31. Classes are better than modules when
    1. One needs multiple instances
    2. One needs static instantiation of opaque bindings
    3. Never. They are just different ways to do the same things
    4. One need single instance functional subdivision
32. External fragmentation of a heap is when an allocation request is combined with another request for increased efficiency
    1. True
    2. False
33. In C, having + represent both integer and floating point addition is an example of
    1. Operator overloading
    2. Polymorphism
    3. Aliasing
    4. Dynamic method dispatch
34. The C++ using statement can be dangerous because
    1. It violates the information hiding security model
    2. It always exposes all names in a namespace
    3. It can be misused accidentally to make more names visible than desired
    4. B and c
35. Once a binding is created between a name and an object, it persists until both the name and the object are destroyed
    1. False
    2. True
36. The difference between modules and classes is that
    1. There is no difference. They are just difference names for the same concept
    2. They both have inheritance but modules are more efficient to implement
    3. Modules have inheritance and dynamic name binding and class don’t
    4. Classes have inheritance and dynamic methods dispatch and modules don’t
37. All programming languages with nested scopes provide a scope resolution operator
    1. True
    2. False
38. Reference counting, Mark/sweep, copying, and generational are kinds of
    1. Lexical parsing methods
    2. Garbage collection
    3. Dynamic binding strategies
    4. Regular expression analysis
39. The process by which a compiler automatically converts a value from one type to another is called
    1. Operator overloading
    2. Polymorphism
    3. Aliasing
    4. Coercion
40. External fragmentation of a heap is when there is enough space, but it is not in one block
    1. False
    2. true
41. The three lifetime spans are
    1. Compile time, run time, and binding time
    2. Dynamic, static, and lexical
    3. Preallocated, postallocated and median-allocated
    4. Static stack, and heap
42. A symbol table must maintain information about only one binding for a given name
    1. True
    2. False
43. A variable of an opaque types
    1. May not be otherwise manipulated
    2. May be passed as an argument to the defining module
    3. May be declared
    4. A and c
    5. A, b, c
44. For an object to be used, it must be reachable by
    1. Being a global object
    2. Existing in an active stack frame
    3. Being pointed to by a reachable object
    4. B or c
    5. A b or c
45. Polymorphism allows
    1. A reduction in execution time due to increased stack efficiency
    2. Multiple classes to be grouped in an inheritance hierarchy
    3. A single subroutine to accept arguments of multiple types
    4. Dynamic reallocation of memory as the heap change size.
46. Association lists become inefficient when
    1. Bindings are overridden for local variables
    2. The global namespace is fragmented
    3. The heap and the stack are both used for memory allocation
    4. Programs get larger and more complex
47. Aliasing can cause inefficient code because
    1. The heap will become fragmented
    2. The stack has to be realigned after each reference
    3. The compiler cannot depend on values kept in the registers
    4. Multiple names must be checked for each memory reference
    5. A, b, c and d
48. In pascal and C, all data must have names
    1. False
    2. True
49. Object lifetimes and name lifetimes do not have to be the same
    1. False
    2. True
50. Aliasing is
    1. When a scope is entered and new bindings are created for local variables
    2. When a name can refer to two or more objects
    3. When an object can be referred by two or more names
    4. When a scope is exited and the binding associated with the scope are popped off
51. If a language permits recursion
    1. Heap allocation must be used for both local and global variables
    2. Static allocation of local variables may not be used
    3. Dynamic allocation of global variables may not be used
    4. Stack allocation must be used for global variables
52. Which of the following is/are not known until runtime?
    1. Explicitly allocated object sizes
    2. Recursion limits
    3. Higher-order functions construction
    4. A, b and C
53. There is always a one-to-one correspondence between names and objects
    1. False
    2. True
54. In dynamic scoping, two common ways to keep track of bindings are …
    1. Association lists and central reference tables
    2. Central association lists and dynamic symbol tables
    3. Dynamic formal lists and static table references
    4. Set-theoretic closures and Cohen-Sutherland indexing
55. An association list (A-list) is
    1. A list of linked list
    2. A linked list
    3. A list of key, value pairs
    4. A, b and c
56. In dynamic scoping, the bindings are
    1. Resolved at compile time for efficiency
    2. Dependent exclusively on the lexical structure of the program
    3. Dependent on calling sequences
    4. Formalized at link time using context-free grammar rules
57. A binding to an object that is no longer alive is called a \_\_\_\_\_
    1. Dead variable
    2. Dead pool
    3. Dangling reference
    4. Garbage pointer
58. A binding’s scope is
    1. The method by which the binding is allocated
    2. Created at compile time in a Chomsky-hierarchy language
    3. Used to reduce the chance of semantic error
    4. The part of the program where it is active
    5. A, b, c and d
59. Why are central reference tables preferred to association lists?
    1. Multiple values can be associated with each name
    2. Lookup is fast because each slot is a linked list with the most recent binding in front
    3. When a scope is exited, the bindings associated with the scope are popped off
    4. When a scope is entered, each new binding is pushed onto the front of the list for the name
    5. A, b, c, and d
60. Modules into which name must be imported explicitly are called
    1. Static scopes
    2. Closed scopes
    3. Open scopes
    4. Opaque scopes
61. Subroutines are \_\_\_\_\_ abstractions and classes are \_\_\_\_\_\_\_\_\_ abstractions
    1. Control ///// data
    2. Formal ///// theoretical
    3. Data ////// control
    4. Operative ////// semantic