---

1. **Uniform symbol table**
   a. Contains all constants in program
   b. Is a permanent table of decision rules in the form of patterns for matching with the uniform symbol table to discover syntactic structure
   c. Consists of full or partial list of tokens as they appear in program created by lexical analysis and used for syntax analysis and interpretation
   d. A permanent table which lists all key words and special symbols of language in symbolic form

2. **When inserted the characters of the string: KRPCSNYTJM into hash table of size 10 by using hash function H (X) = (ORD (X)-ORD ('A') + 1) mod 10 and linear probing to resolve collisions. Insertions that cause collisions are**
   a. C, S
   b. Y, M
   c. J, M
   d. S, T

3. **A hash function is defined as F(key) = key mod 7 with linear probing is used to insert the keys 37,38,72,48,98,11,56 into a hash table indexed from 0 to 6 .Then in which location the key 11 is stored.**
   a. 6
   b. 5
   c. 1
   d. 4

4. **A hash function is defined as F (key) = key mod 7 with linear probing is used to insert the keys 37,38,72,48,98,11,56 into a hash table indexed from 0 to 6 .Then following key is stored as the last element**
   a. 98
   b. 56
   c. 11
   d. 48

5. **The storage strategy in which activation record is maintained even after the execution of a procedure is completed is**
   a. Stack allocation
   b. Heap allocation
   c. Static allocation
   d. Dynamic allocation

6. **The average search length of the table in the linear list of 'n' records for the look up operation requires**
   a. $\frac{n(n+1)}{2}$
   b. $\frac{n+1}{2}$
   c. $\frac{n+2}{2}$
   d. $\frac{n-1}{2}$

7. **In the search trees implementation the expected time required to enter 'n' names and make 'm' enquiries is proportional to**
   a. (n +m) log n
   b. (m +n) log m
   c. m n log n
   d. m n log m

8. **Time is independent of number of entries in a symbol table among**
   a. Linear list
   b. Search tree
   c. Hash table
   d. Self-organizing lists

9. **Based on the property of locality of reference the symbol table is implemented in**
   a. Linear list
   b. search tree

c. hash table
d. self-organizing list

10. Access time of symbol table will be logarithmic if it is implemented by
a. linear list
b. search tree
c. hash table
d. self-organizing list

11. The following symbol table implementation has the minimum access time.
a. hash table
b. search tree
c. linear list
d. self-organizing list

12. A data structure which is used to store information about various source languages constructs is
a. parse tree
b. synthesized grammar
c. symbol table
d. derivation tree

13. The names referenced frequently will be at the front among the following implementation of the symbol table
a. search trees
b. hash tables
c. self-organizing lists
d. linear lists

14. Requirement of storage for an organization using the variable length entries to the fixed length entries is
a. less
b. more
c. equal
d. can't compare

15. In linked list, to insert 'n' names and 'm' enquiries in the symbol table, the total work is
a. c n m
b. m + n
c. c n (n + m)
d. c m (n + m)

16. On an average, the time required per operation to insert 'n' names and 'm' enquiries in the symbol table by hashing requires
a. c n (n + m)
b. c m (n + m)
c. constant
d. no

17. The following allocation can become stack allocation by using relative address for storage in activation records
a. Static
b. Heap
c. Dynamic
d. Hash

18. The allocation strategy in which only one occurrence of each object is allowable at a given moment during program execution is
a. Stack
b. Static
c. Register
d. Heap

19. During compile time it is impossible to determine the number of times recursive procedure is going to be involved in the following allocation strategy
a. Stack
b. Static
c. Register
d. Heap

20. The allocation in which the activation record is fixed at compile time in memory is
- **a. Static**
- b. Dynamic
- c. Heap
- d. Stack

21. The following is not possible in static storage schema
- a. Fixed length strings
- **b. Nested procedures**
- c. Switch statements
- d. Structures

22. The allocation strategy in which if names are bounded, then there is no need for a runtime support package is
- a. Stack
- **b. Static**
- c. Register
- d. Heap

23. The allocation which manages storage for all data objects at compile time is
- a. Dynamic
- **b. Static**
- c. Heap
- d. Stack

24. The allocation strategy in which the size of each object must be known at compile time is
- **a. Static**
- b. Stack
- c. Register
- d. Heap

25. FORTAN has the following storage allocation strategy
- a. Stack
- **b. Static**
- c. Heap
- d. Register

26. In the following memory allocation program variables remain permanently allocated irrespective of their accessibility at any stage of programs execution.
- a. Stack
- **b. Static**
- c. Heap
- d. Register

27. Data structures cannot be created dynamically in the following allocation
- **a. Static**
- b. Stack
- c. Heap
- d. Register

28. Language that does not support dynamic allocation is
- a. ALGOL
- **b. FORTAN**
- c. PASCAL
- d. PL/I

29. Dynamic allocation of storage areas with VSAM files is accomplished by
- a. Hashing
- **b. Control splits**
- c. Overflow areas
- d. Relative recording

30. The following allocation can become stack allocation by using relative address for storage in activation records.
- **a. Static**
- b. Heap
- c. Dynamic
- d. Hash

31. The allocation in which the array base address is not known at compile time.
a. Static
b. Dynamic
c. Register
d. Paged

32. The storage allocation followed for Strings in SNOBOL is.
a. Stack
b. Heap
c. Register
d. Static

33. In stack allocation a new activation record is pushed into the stack for each exception of
a. Data base
b. Assignment statement
c. Procedure
d. MACRO

34. When the procedure ends in stack allocation the record is
a. Pushed
b. Popped
c. Peeped
d. Displayed

35. When there is reference to storage that has been allocated, then this may occur
a. Dangling
b. Indexed
c. Offset
d. Virtual

36. The allocation which is useful for implementing data whose size varies as the program is running is
a. Stacks
b. Static
c. Heap
d. Register

37. The following manages runtime storage as stack
a. Static
b. Dynamic
c. Heap
d. Stack

38. The allocation strategy which allocates and deallocates storage as needed at runtime from heap data area is
a. Static
b. Stack
c. Heap
d. Register

39. Heap allocation is required for languages that
a. support recursion
b. support dynamic data structure
c. use dynamic scope rules
d. use dynamic programming

40. In the following allocation words in the activation record can be accessed as offsets from the value in the register.
a. Static
b. Dynamic
c. Heap
d. Stack

41. Storage for names local to the procedure appears in
a. Activation record
b. Symbol table
c. Hash table
d. Process table

42. The call statement in the intermediate code is implemented by

a. MOV
b. GOTO
c. MOV and GOTO
d. HALT

43. The allocation which is useful for handling recursive procedure is

a. Stack
b. Heap
c. Static
d. Register

44. A record in PASCAL is defined by

```
type rec = record
- - - - - - - - - - - - - - n:integer;
- - - - - - - - - - - - - - case(var1,var2) of
- - - - - - - - - - - - - - - - - - - - - - var1:(x,y:integer);
- - - - - - - - - - - - - - - - - - - - - - var2:(p,q:real);
- - - - - - - - - - - - - - end;
end;
```

Suppose an array of 100 such records was declared as a machine which uses 4 bytes for an integer and 8 bytes for a real. How much space would the compiler has to reserve for an array?

a. 1600
b. 1800
c. 2000
d. 2020

45. The program fragment that follows is written in a block structured language .Assuming that it is syntactically correct and determine its output

```
begin
- - - - - integer x,y;
- - - - - - - - - - x:=3;
- - - - - - - - - - y:=7;
begin
- - - - - integer x;
begin
- - - - - integer y;
- - - - - - - - y:=9;
- - - - - - - - x:=2*y;
end;
- - - - - x:=x+y;
- - - - - print(x);
end;
- - - - - print(x);
end.
```

a. 3 - - - 25
b. 25 - - - 3
c. 3 - - - 3
d. 25 - - - 25

46. Assuming 32 bit 2's complement representation for integers and 8-bit ASCII representation for characters, how many bytes of memory are occupied by the following Pascal declaration

```
var rec: RECORD
- - - - - - - - - - - - F1:RECORD
- - - - - - - - - - - - - - - - part1:RECORD
- - - - - - - - - - - - - - - - - - - - - - p1:char;
- - - - - - - - - - - - - - - - - p2:integer;
- - - - - - - - - - - - - - - - - END;
- - - - - - - - - - - - - - - - - - - - - - part2 :char;
F2: array [1..2] of RECORD
- - - - - - - - - - - - - - - - - - - - - - x1: integer;
- - - - - - - - - - - - - - - - - - - - - - x2: array [1..3] of char;
- - - - - - - - - - - - - END;
- - - - - END;
```

Total allocated memory is

a. 10
b. 20

c. 30
d. 40

47. **struct**
{
short s[5];
union{
float y;
long z;
} u;
}]t;
Assume short, float, long occupy 2 bytes,4 bytes,8 bytes respectively .The storage for variable 't' ignoring alignment is
a. 22 bytes
b. 14 bytes
c. 18 bytes
d. 10 bytes

48. **The output of following program in a language that has dynamic scoping is:**
var x:real;
procedure s1;
begin
, , , , , , , , , , , , , , , print(x);
end;
procedure s2;
var x:real;
begin
x:=0.125;
s1;
end;
begin
x:=0.25;
s1;
s2;
end;
a. 0.125 - - 0.125
b. 0.25 - - 0.125
c. 0.25 - - 0.25
d. 0.125 - - 0.25

49. **The program fragment that follows is written in a block structured language Assuming that it is syntactically correct and determine its output**
union{
struct
{
char a,b,c,d;
}s1;
struct
{
int i ,j;
short s;
} s2; long
z; float f;
double m;
} u;
The size of u is
a. 8 bytes
b. 4 bytes
c. 2 bytes
d. 16 bytes

50. **If the language has dynamic scoping and parameters are passed by reference, then what will be printed by the following?**
p( )
var n:int;
procedure w (var x:int)

```
begin
x=x+1;
print(x);
end;
procedure D( )
begin
var x:int;
x=3;
w(n);
end;
begin /* main program*/
n=10;
D;
end.
```

a. 10
b. 11
c. 3
d. 4

51. Consider the following variant record declaration in Pascal

```
Type abc =RECORD
- - - - - - - - - - - - - - - - - - x:integer
- - - - - - - - - - - - - - - - - - - - - - - - case(y:integer) of
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 1:(m:integer,n:real);
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 2:(e,f:integer);
END;
```

suppose a program uses an array of p such records .Integer needs 2 bytes of storage and real needs 4 bytes of storage. If an array occupies 480 bytes, the value of p is

a. 10
b. 64
c. 60
d. 80

52. A region of validity every name possesses in the source program is

a. Scope
b. Life time
c. Binding
d. Domain

53. A following for the procedure is a number that is obtained by standing with a value of one for the main and adding one to it every time we go from an enclosing to enclosed procedure.

a. Nesting loop
b. Nesting depth
c. line numbering
d. Recursion

54. Consider the following block of code

```
p( )
begin
x=10, y=3;
func (y,x,x);
print (x,y);
end;
func (x,y,z)
begin
y=y+4;
z=x+y+z;
end;
```

If parameters are passed by reference ,the output of above procedure is

a. 31 - - 3
b. 3 - - 31
c. 3 - - 3
d. 31 - - 31

55. In analyzing the compilation of a program 'machine independent optimization' is associated with

a. Recognition of basic syntactic construction through reduction
b. Recognition of basic elements and creation of uniform symbols

c. Creation of more optimal matrix
d. Use of macro processor to produce more optimal assembly code

56. The following is used as the key to accessing the scope information from the symbol table
a. Procedure name
b. Procedure name, nesting depth
c. Nesting depth
d. Usage count

57. The following of a procedure is a number that is obtained by standing with a value of one for the main and adding one to it every time we go from an enclosing to enclosed procedure
a. Nesting loop
b. Nesting depth
c. Nesting breadth
d. Recursion

58. Generation of intermediate code based on a abstract machine model is useful in Compilers because
a. It makes implementation of lexical analysis and syntax analysis easier
b. Syntax directed translations can be written for intermediate code generation
c. It enhances the portability of front end of compiler
d. It is not possible to generate code for real machines directly from high level Language programs

59. An optimized compiler
a. Is optimized to occupy less space
b. Is optimized to take less time for execution
c. Optimized the code
d. Optimized to small typing font

60. The following refers to the techniques a compiler can employ in an attempt to produce a better object language program than the most obvious for a given source program is
a. Code generation
b. Code optimization
c. Code execution
d. Code debugging

61. The "90-10" rule states that
a. 90 % of code is executed in 10 % of time
b. 90 % of time is spent in 10 % of code
c. 90 % of time is spent in correcting 10 % of errors
d. 10 % of time is spent in correcting 90 % of errors

62. The most heavily travelled parts of programs are
a. Inner loops
b. Constants
c. Static variables
d. Global variables

63. Machine independent code optimization can be applied to
a. Source code
b. Intermediate representation
c. Object code
d. Run-time output

64. At a point in a program if the value of variable can be used subsequently, then that variable is
a. Live
b. Dead
c. Duplicate
d. Aliasing

65. The region of validity every name possesses in the source program is
a. Scope
b. Life time
c. Binding
d. Domain

66. Which of the following is not a peephole optimization?
a. Removal of unreachable code
b. Elimination of multiple groups

c. Elimination of loop invariant compilation
d. Loop unrolling

67. At point p if no matter what path is taken from p, the expression e will be evaluated before any of its operands are defined, then expression e is said to be
a. Lazy
b. Late
c. Busy
d. Dummy

68. The evaluation which avoids the action at all which is clearly advantageous
a. Lat
b. Lazy
c. Critical
d. Smart

69. Which of the following comments about peep hole optimization are false.
a. It is applied to small part of the code
b. It can be used to optimize intermediate code
c. To get the best out of this, it has to be applied repeatedly
d. It can be applied to the code that is contiguous

70. The following examines short sequences of code and determines a sequence can be replaced by a shorter equivalent sequence.
a. Peep hole optimizer
b. Assembler
c. Linker
d. Loader

71. The evaluation ordering in which we know beforehand that will have to perform the action anyway, but we find it advantageous to perform it as late as possible
a. Late
b. Lazy
c. Critical
d. Smart

72. The evaluation ordering in which, code for node is issued as soon as the code for all of its Operands has been issued
a. Early
b. Late
c. Lazy
d. Critical

73. Peep-hole optimization is form of
a. Local optimization
b. Constant folding
c. Copy propagation
d. Data flow analysis

74. If the transformation of a program can be performed by looking only at statements in basic block then it is said to be
a. Local
b. Global
c. Algebraic
d. Matrix

75. If E was previously computed and the values of variable in E have not changed since previous computation ,then an occurrence of an expression E is
a. Copy propagation
b. Dead code
c. Common sub expressions
d. Constant folding

76. In block B if s occurs in B and there is no subsequent assignment to y within B, then the copy statement s:x :=y is
a. Generated
b. Killed
c. Blocked

d. Dead

77. In block B if x or y is assigned there and s is not in B then s: x :=y is
a. Generated
b. Killed
c. Blocked
d. Dead

78. If two or more expressions denote same memory address, then expressions are
a. Aliases
b. Definitions
c. Superior
d. Inferior

79. Operations that can be removed completely are called
a. Strength reduction
b. Null sequences
c. Arithmetic simplification
d. Constant folding

80. Given the following code
X=A+B;
100 Y=A+B;
And the corresponding optimized code as
- - - - - - - - - - Z=A+B;
- - - - - - - - - - X=Z;
- - - - - - - - - - 100 Y=Z;
When will be optimized code pose a problem?
a. Z may not remain same after the control reaches here first time since it is labeled
b. When Z is undefined
c. When memory is consideration
d. Doesn't pose a problem

81. Can the loop invariant X = A - B from the following code be moved out
For I = 1 to 10
A = B * C;
X = A - B;
a. Yes
b. No
c. X=A-B is not invariant
d. Wrong code

82. The following transformation takes an expression that yields the same result independent of the number of times.
a. Reduction in strength
b. Induction variable
c. Constant folding
d. Code motion

83. The optimization that avoids a test at each iteration is
a. Loop unrolling
b. Loop ramming
c. Loop nesting
d. Loop scrolling

84. The method that merges the bodies of two loops is
a. Loop nesting
b. Constant folding
c. Loop ramming
d. Loop unrolling

85. The code for repetition can be optimized by
a. Loop unrolling
b. Loop jamming
c. Nested loop
d. Code motion

86. The following can not be used to identify loops
a. Depth first ordering
b. Reducible graphs
c. Dominators
d. Flow chart

87. Replacement of a string concatenation operator "||" by an addition is example for
a. Strength reduction
b. Operator overriding
c. Operator over loading
d. Operator elimination

88. The replacement of an expensive operation by a cheaper one
a. Operator overriding
b. Operator overloading
c. Reduction in strength
d. Operator elimination

89. The loop which does not contain any other loops are
a. Natural loop
b. Inner loop
c. Main loop
d. Nested loop

90. Identify the basic blocks in the following code
10 goto 20
20 goto 10
a. Go to 20
b. Go to 10
c. Two independent basic blocks
d. Both statements in one block

91. Loop is collection of nodes that
a. Is loosely connected
b. Is strongly connected with several entries
c. Is having several entries
d. Is strongly connected with unique entry

92. In loop optimization technique a loop whose body is rarely executed is
a. Dead code
b. Blank code
c. Redundant code
d. Blank stripper

93. If the following optimization is attempted, count of number of jumps to each label can be found from symbol table of that label
a. Back tracking
b. Peephole optimization
c. Dynamic programming
d. Global optimization

94. The use of the following greatly improves the code when pushing or popping a stack, as in parameter passing
a. Flow of control
b. Auto increment addressing modes
c. Auto decrement addressing modes
d. Both auto increment and auto decrement addressing modes

95. The following can be done through peephole optimization
a. Dynamic programming
b. Greedy method
c. Unreachable code clinination
d. Code generation

96. The following algebraic identities can be eliminated through peephole optimization
a. x:=x+0
b. x:=x*1
c. x:=x+0 and x:=x*1

d. Jumps over jumps

97. Replacing the exponent operator by the shift of multiplication is
a. Constant folding
b. Reduction in strength
c. Copy propagation
d. Elimination of dead variables

98. Method to improve the target program by examining a short sequence of target instructions
a. Back tracking
b. Peephole optimization
c. Dynamic programming
d. Global optimization

99. Program transformation that is not characteristic of peephole optimization
a. Redundant instruction elimination
b. Flow of control optimization
c. Back patching
d. Use of machine idioms

100. The following is the small moving window in target program
a. Screen saver
b. Graphics
c. Peephole
d. Greedy window

101. Use of machine idioms is one of the characteristic of
a. Back tracking
b. Peephole optimization
c. Dynamic programming
d. Global optimization

102. Peephole optimization is to eliminate
a. Constant folding
b. Jumps over jumps
c. Copy propagation
d. Elimination of dead variables

103. Machine independent optimization is
a. Register allocation
b. Frequency reduction
c. Data intermixed with instructions
d. Machine features instructions

104. Computing two or more identical computations to a place in the program where the computation can be done once and the result to used in the entire original places is
a. Replacing
b. Profiling
c. Hoisting
d. Rearranging

105. Deducing at compile time that the value of an expression is a constant and using that constant instead is
a. Constant folding
b. Constant deduction
c. Macro substitution
d. Copy propagation

106. The following computation is done and placed the expression before the loop in the transformation of code motion
a. Loop-invariant
b. Overloading
c. Overriding
d. Constant folding

107. Changing the form to preserve the program semantics is
a. Coding
b. Frequency reduction
c. Constant folding

d. Constant substitution

108. Replacing expressions by their value if the value can be computed at compile time is called
a. Constant substitution
b. Constant folding
c. Macro
d. Infix expression

109. Replacing the expression 2*3.14 by 6.28 is
a. Constant folding
b. Induction variable
c. Strength reduction
d. Code reduction

110. An estimate of how frequently a variable used is
a. Usage count
b. Reference count
c. Program count
d. Process count

111. The modification that decreases the amount of code in a loop is
a. Constant folding
b. Constant reduction
c. Code motion
d. Reduction strength

112. Movement of the code from inside to outside is
a. Coding
b. Frequency reduction
c. Constant folding
d. Constant substitution

113. In DAG the interior nodes are labeled with
a. Numbers in DFS
b. Numbers in BFS
c. Identifiers
d. Special colors

114. Sorting techniques used for evaluation of interior nodes of DAG in any order is
a. Quick sort
b. Selection sort
c. Merge sort
d. Topological sort

115. If every path from the initial node goes through that particular node, then the node is said
a. Header
b. Dominator
c. Parent
d. Descendant

116. DAG has
a. Only one root
b. Any number of roots
c. No root
d. No nodes at all

117. A basic block can be analyzed by
a. Flow graph
b. Flow chart
c. DAG
d. RAG

118. In DAG, interior nodes are labeled by
a. Operator symbol
b. Unique identifiers
c. Operands
d. Node numbers

119. Common sub expressions can be detected automatically by using

a. Flow graph
b. Flow chart
c. DAG
d. RAG

120. Date structures useful for implementing transformations on basic blocks are
a. RAG
b. DAG
c. Gaunt chart
d. B^+ tree

121. DAG representation of a basic block allows
a. Automatic detection of local common sub expression
b. Automatic detection of induction variables
c. Automatic detection of loop invariables
d. Automatic motion to code

122. In DAG, labeling of nodes are
a. Mandatory
b. Optional
c. Compulsory
d. Automatically generated

123. The statement that may be safely removed without changing the value of basic block is
a. Live code
b. Dead code
c. Temporary variables
d. Common sub-expression

124. The block which can be transformed into an equivalent block in which each statement that defines a temporary defines a new temporary is
a. Normal form
b. Temporary block
c. Procedural block
d. Null block

125. The following basic block permits all statement interchanges that are possible
a. Normal form
b. Dead block
c. Temporary block
d. Null block

126. Determine the number of definition and reference points of variable A in the following code
A = Z; /* statement S0*/
A = B; /* statement S1*/
C = D; /* statement S2*/
C = C + A;
a. 1,3
b. 2,3
c. 0,4
d. Can't be determined

127. Determine the definition points d; affecting the expression C = C+A
In the following code
A=Z; /* statement S0*/
A=B; /* statement S1*/
C=D; /* statement S2*/
C=C+A;
a. Statement S0 and S1
b. Statement S1 and S2
c. Statement S0 and S2
d. Undetermined

128. Structure preserving transformations on basic blocks are
a. Dead code elimination
b. Strength reduction
c. Copy propagation
d. Constant folding

129. In reducible flow graph, the following edges consists only of edges whose heads dominate their tails.
a. Forward edges
b. Back edges
c. Pre-header
d. Header

130. Determine the pre-dominant block of block B2 in the program flow graph from the following code
../* Block B0*/
10 go to 100 /* Block B1*/
100 go to 10 /* Block B2*/
a. B0
b. B1
c. B2
d. Code insufficient

131. A sequence of consecutive statements in which the flow of control enters at the beginning and leaves at the end without halt or possibility of branching except at the end is
a. Flow graph
b. DAG
c. Basic block
d. Loop

132. The first statement in basic block is
a. Leader
b. Header
c. Main statement
d. Follow

133. Any statement that immediately follows the following statement is a leader
a. First statement
b. GOTO or conditional GOTO
c. While loop
d. Array declaration

134. The following among these is not structure preserving transformations on basic blocks
a. Common sub-expression elimination
b. Invariant variables elimination
c. Dead code elimination
d. Interchanging of two independent adjacent statements

135. Determine the path(s) in the program flow graph for the code
10 goto 100 /* Block B1*/
100 goto 10 /* Block B2*/
a. (B1, B2)
b. (B2,B1)
c. (B1,B2,B1)
d. No path

136. The graph that shows the basic blocks and their success of relationship is called
a. DAG
b. Flow graph
c. Control graph
d. Hamilton graph

137. Which of the following class of statement usually produces no executable code when compiled.
a. Declaration
b. Assignment
c. Input and output
d. Structural statements

138. Edges in the flow graph whose heads dominate their tails are called
a. Back edges
b. Front edges
c. Flow edges
d. Head edges

139. Ambiguous definitions of x are

a. An assignment through a pointer that could refer x
b. Assignment to x
c. Storing value in x
d. Reading a value from an I/O device

140. The reaching definition information is stored in
a. Activation record
b. Symbol table
c. Ud-chains
d. Du-chains

141. Data flow equation has the form
a. OUT[S] =GEN[S] ∩ (IN [S] + KILL [S])
b. OUT[S] =GEN[S] ∩ (IN [S] -KILL [S])
c. OUT[S] =GEN[S] (IN [S] + KILL [S])
d. OUT[S] =GEN[S] (IN [S] -KILL [S])

142. A call of a procedure with c as a parameter is
a. Unambiguous definition
b. Ambiguous definition
c. Global definition
d. Function definition

143. An inherited attribute among these is
a. IN
b. OUT
c. KILL
d. GEN

144. A synthesized attribute depending on IN is
a. OUT
b. IN
c. KILL
d. GEN

145. The following of a variable x is a statement that assigns, or may assign a value to x
a. Definition
b. Memory allocation
c. Scope
d. Type binding

146. The following is a special case of region that is strongly connected and includes all its back edges into the header
a. Tree
b. Loop
c. Forest
d. Branch

147. Local and loop optimization in turn provide motivation for
a. Data flow analysis
b. Constant folding
c. Peep hole optimization
d. DFA and constant folding

148. A half -way automation of full symbolic interpretation ,in which the stack representation is replaced by a collection of sets are
a. Flow graph
b. Flow chart
c. Data flow equations
d. Data dictionary

149. The ontain the following items removed by the node N
a. IN (N)
b. OUT (N)
c. KILL (N)
d. GEN (N)

150. The following contains the items added by the node N
a. IN (N)

b. OUT (N)
c. KILL (N)
d. GEN (N)

151. Given the following code:
Procedure p(a, b, c)
z=a + b;
c=.;
x= a + b;
in which case a + b cannot be treated as common sub expression?
a. No such case exists
b. Cannot be determined here
c. a + b is not an expression
d. call by reference for p(x, y, z)

152. Find the common sub expression in the following code:
declare A, C,D;
A=C+D;
D=D-1;
D=C+D;
a. C+D
b. D+C
c. No such expression
d. D

153. Common expressions can be obtained from
a. DAG
b. RAG
c. TREE
d. Syntax tree

154. The value number of A[i] and B for sub expression A[i] * B in the following code will be
X=A[i]*B;
Y=A[i]*B;
a. Undefined
b. Same
c. Different
d. Code insufficient to determine

155. The following basic block contains
a:=b + c
b:=a-d
c:=b+c
d:=a-d
a. Two common sub expressions
b. Only one common sub expression
c. Dead code
d. Temporary variable

156. In the following basic block
a:=b + c
b:=a-d
c:=b+c
d:=a-d
The value of b in the third statement is
a. Same as b in first statement
b. Different from b in first statement
c. 0
d. 1

157. The following nodes will correspond to nodes with more than one panet in common subexpression are
a. Shared nodes
b. Labeled nodes
c. Unlabeled nodes
d. Interior nodes

158. Common sub expression in the following code

Declare A,B ,pointer P;
P=address(A)
A=content(P)+B;
B=content(P)+B;

a. No common sub expression
b. A=content(P)+B;
c. B=content(P)+B;
d. A+B

159. Given that a quadruple may be allowed to have one non-program variable as operand .Does there exists a set of quadruple that will not identify the common sub expression X * Y in the following code?
A=X * Y; - - - - - - B= W * Z * X * Y;

a. Yes
b. No
c. X= A-B is not loop invariant
d. A

160. In the source code
X= a*a + 2*a*b + b*b;
Y=a*a-2*a*b + b*b; contains number of common sub expressions

a. 3
b. 2
c. 6
d. 0

161. Find the induction variables from the following code
A = -0.2;
B = A+5.0;

a. A
b. B
c. Both A and B are induction variables
d. No induction variables

162. A loop variable

a. May not be induction variable if it is allowable to change the value of the loop variable inside the loop.
b. Copy propagation variable
c. Frequency reduction operator
d. Jumping statement

163. The following is not a function preserving transformation

a. Copy propagation
b. Constant folding
c. Elimination of induction variables
d. Dead-code elimination

164. When there are two or more induction variables in a loop we have an opportunity to get rid of all but one, and this process is called as

a. Loop optimization
b. Induction variable elimination
c. Loop induction
d. Induction variable generation

165. Find the induction variables in the following code of the loop
n:=100;
for i:=1 to 10 do
begin
j:=i*8+k;
n=n-10;
end;

a. i ,n
b. Only i
c. Only n
d. No induction variables

166. The following variable is any ordinal variable whose value follows the control variable of a loop in a linear relationship

a. Folding variable
b. Invariant variable
c. Strength variable
d. Static variable

167. The optimization technique which is typically applied on loops
a. Peep hole optimization
b. Constant folding
c. Removal of invariant computations
d. Dead code elimination

168. If every time the variable x changes values in a loop L then variable x is called
a. Assignment
b. Subscript
c. Induction variable
d. Strength

169. If a variable X of a loop L changes the values every time it is incremented or decremented by some constant then that variable is
a. Induction variable
b. Loop variable
c. Control variable
d. Invariant variable

170. Induction variable elimination is important technique used in
a. Local optimization
b. Loop optimization
c. Global optimization
d. Code generation

171. The analysis that cannot be implemented by forward operating data flow equations mechanism is
a. Interprocedural
b. Procedural
c. Live variable analysis
d. Data

172. If m has a Parent which has not yet been evaluated then m is defined as
a. Used
b. Live
c. Dead
d. Fuzzy

173. If m value is needed outside the block then m is defined as
a. Used
b. Live
c. Dead
d. Fuzzy

174. Live variables coming into the block can be defined as
a. in [B] = use [B] (out [B] - def [B])
b. in [B] = use [B] ∩ (out [B] - def [B])
c. in [B] = use [B] (out [B] + def [B])
d. in [B] = use [B] ∩ (out [B] + def[B])

175. The data flow analysis can obtains information about the IN and INOUT parameters of R, and carried into the routine by symbolic interpretation is
a. Interprocedural
b. Procedural
c. Live variable analysis
d. Static

176. In statement s if r-value is required, then that variables is
a. Used
b. Live
c. Dead
d. Fuzzy

177. If the information from live analysis is available all information can be deleted about a variable from the following description the moment its live range is left is

a. Register variable
b. Symbol table
c. Hash table
d. Address descriptor

178. The following requires information to be propagated backwards along the flow of control, from an assignment to a variable to its last preceding use: the variable is dead in the area in between is
a. Last -def analysis
b. Live analysis
c. Data analysis
d. Data flow

179. Live variables coming out of the block can be defined as
a. $\text{Out}[B] = \bigcup_{S \text{ a successor of } B} in[S]$
b. $\text{Out}[B] = \bigcap_{S \text{ a successor of } B} in[S]$
c. $\text{Out}[B] = \bigcup_{S \text{ a successor of } B} def[S]$
d. $\text{Out}[B] = \bigcap_{S \text{ a successor of } B} def[S]$

180. The following problem is to compute for a point p the set of uses s of a variable x such that there is a path p to s that does not redefine x is
a. ud-chaining
b. du-chaining
c. Spanning
d. Searching

181. Important use of live-variable information comes
a. During source code
b. During construction of flow graph
c. During generation of object code
d. During error correction

182. If $c-gen[B]$ to be set of all copies generated in block B and $c-kill[B]$ the set of copies in U that are killed in B then $out[B]$ =
a. $\text{c}-\text{gen}[\text{B}] \cup (\text{in}[\text{B}] - \text{c} - \text{kill}[\text{B}])$
b. $\text{c}-\text{gen}[\text{B}] \cup (\text{in}[\text{B}] + \text{c} - \text{kill}[\text{B}])$
c. $\text{c}-\text{gen}[\text{B}] \cap (\text{in}[\text{B}] - \text{c} - \text{kill}[\text{B}])$
d. $\text{c}-\text{gen}[\text{B}] \cap (\text{in}[\text{B}] - \text{c} - \text{kill}[\text{B}])$

183. If $c-gen[B]$ to be set of all copies generated in block B and $c-kill[B]$ the set of copies in U that are killed in B then $in[B]$ if B not initial is
a. $\bigcap_{P \text{ a predessor } B} out[P]$
b. $\bigcup_{P \text{ a predessor } B} out[P]$
c. $\bigcap_{P \text{ a predessor } B} out[P] - c-gen\ [B]$
d. $c-gen\ [B] + \bigcap_{P \text{ a predessor } B} out[P]$

184. Full optimized code after using variable propagation technique
A=B*C, D=C,X=B*D+E
a. A=B*C ,D=C,X=B*C+E b.
A=B*C ,D=C,X=B*D+C **c.**
A=B*C ,X=A+E
d. A=B*C ,D=C,C=B*D+E

185. If $c-gen[B]$ to be set of all copies generated in block B and $c-kill[B]$ the set of copies in U that are killed in B then $in[B]$ if B initial is
a. $\emptyset$
b. out [B]
c. U

d. c-gen[B]

186. The following often turns the copy statement into dead code
a. Copy propagation
b. Dead code
c. Common sub expressions
d. Constant folding

187. Assignment of form f := g is called
a. Constant folding
b. Copy statements
c. Dead code
d. Induction variables

188. Substitute y for x for copy statement s: x: =y if the following condition is met
a. Statement s may be the only definition of x reaching u
b. X is dead
c. Y is dead
d. X and y are aliases

189. In block B if s occurs in B and there is no subsequent assignment to y within B, then the copy statement $s : x := y$ is
a. Generated
b. Killed
c. Blocked
d. Dead

190. In block B if x or y is assigned there and s is not in B then $s : x := y$ is
a. Generated
b. Killed
c. Blocked
d. Dead

191. If two or more expressions denote same memory address, then expressions are
a. Aliases
b. Definitions
c. Superior
d. Inferior

192. FORTAN is the output of the following compiler
a. TRAN
b. PL/I
c. ALTRAN
d. ANSI

193. Relocation bits used by relocating loader are specified by
a. Relocating loader itself
b. Linker
c. Assembler
d. Macro processor

194. Producing the following code as output has the advantage that it can be placed in a fixed location in memory and can be immediately executed.
a. Relocatable machine code
b. Absolute machine code
c. High level language program
d. Assembly machine code

195. Producing the following code as output allows sub program to be compiled separately.
a. Relocatable machine code
b. Absolute machine code
c. High level language program
d. Assembly machine code

196. A self relocatable program is one which
a. Cannot be made to execute in any area of storage other than the one designated for it at the time of its coding or translation
b. Consists of a program and relevant information for its relocation

c. Can itself perform the relocation of its address sensitive portions
d. It is waiting for an event to occur before continuing execution

197. The code generation phase converts the following into a sequence of machine instructions.
a. Target program
b. Source program
c. Intermediate optimize code
d. Input data given

198. The intermediate optimized code can be sequence of
a. Quadruples
b. Target code
c. Source program
d. Binary language

199. Object code
a. Is ready to execute
b. Is the output of compiler but not assembler
c. Must be loaded before execution
d. Must be rewritten before execution

200. A non relocatable program is one which
a. Cannot be made to execute in any area of storage other than the one designated for it at the time of its coding or translation
b. Consists of a program and relevant information for its relocation
c. Can itself perform the relocation of its address sensitive portions
d. It is waiting for an event to occur before continuing execution

201. A relocatable program is one which
a. Cannot be made to execute in any area of storage other than the one designated for it at the time of its coding or translation
b. Consists of a program and relevant information for its relocation
c. Can itself perform the relocation of its address sensitive portions
d. It is waiting for an event to occur before continuing execution

202. In register inference graph nodes are
a. Operators
b. Operands
c. Symbolic register
d. Variables

203. A K-coloring of G' can be extended to a K-coloring of G in register allocation by
a. Assigning a color not assigned to any of its neighbor
b. Assigning a color same as neighbor
c. Assigning with no color
d. Not possible

204. To determine what name is left in register the following is used
a. Symbol table
b. Hash table
c. Register descriptor
d. Address descriptor

205. Strategy for the following allocation is to assign some fixed number of registers to hold the most active values in inner loop
a. Local register allocation
b. Global registers allocation
c. Graph coloring
d. Automatic allocation

206. In order to free up a register, the content of one of the used registers
a. Must be copied to rarely used register
b. Stored into a memory location
c. No other alternative
d. Must be copied to frequently used register

207. Instructions involving only register operands are _ _ _ _ _ than those involving Memory operands.
a. Faster

b. Slower
c. Almost equal
d. Heap

208. The following tries to keep frequently used value in a fixed register throughout a loop in
a. Usage counts
b. Global register allocation
c. Conditional statements
d. Pointer assignment

209. Values in a program should reside in registers in
a. Register assignment
b. Register allocation
c. Register implication
d. Register indexing

210. Following deals with, in which register each value should reside.
a. Register assignment
b. Register allocation
c. Register implication
d. Register indexing

211. The problem of optimal graph coloring is
a. NP-complete
b. NP-HARD
c. Unsolvable
d. Undefined

212. Which of the following statement is false? The running time of a program depends on
a. The way the registers are used
b. The order in which computations are performed
c. The way the addressing modes are used
d. Size of the computer

213. Following is used to determine that the same name is not already in its memory location, and the live variable information to determine whether the name is to be stored.
a. Symbol table
b. Hash table
c. Register descriptor
d. Address descriptor

214. If the information from live analysis is available all information can be deleted about a variable from the following description the moment its live range is left
a. Register variable
b. Symbol table
c. Hash table
d. Address descriptor

215. Stack machine code is
a. Graphical representation
b. Virtual machine code
c. Three address representation
d. Linear representation

216. Variable descriptors are also known as
a. Register descriptor
b. Address variables
c. Pseudo registers
d. Constants

217. An intuitive garbage collection algorithm that records in each chunk of pointers that point to it is
a. Process count
b. Reference count
c. Record count
d. Program counter

218. The final phase of a compiler is
a. Lexical analyzer
b. Semantic analyzer

c. Code optimizer
d. Code generator

219. The output of code generator is
a. Intermediate code
b. Source code
c. Target code
d. DAG

220. Addressing modes involving registers have
a. Zero cost
b. Cost one
c. Cost two
d. Cost three

221. In IBM Systems/370 machines integer multiplication and division involves
a. Stack
b. Register pairs
c. Heap
d. Arrays

222. Tree translation scheme is
a. Set of tree rewriting rules
b. Set of all traversals
c. Translating all the operations of the tree
d. Translations of trees into arrays

223. The following approach causes a larger number of operations to be performed with a small machine instruction
a. Maximum munch
b. Shift reduce parser
c. Syntax directed translation
d. Dynamic programming

224. In labeling algorithm, that n is a binary node and its children have l_1 and l_2, LABEL (n) if $l_1 = l_2$
a. L_1-1
b. l_2+1
c. $l_1 + l_1$
d. l_1 +1

225. From DAG's the optimal code generation from a tree is also useful when the intermediate code is
a. Parse tree
b. Flow chart
c. Flow graph
d. RAG

226. The input to the code generator is a
a. Sequence of tree at lexical level
b. Sequence of tree at semantic level
c. Sequence of assembly language instructions
d. Sequence of machine idioms

227. The advantage of generating a code from DAG
a. Easily see how to rearrange the order of the final computation sequence
b. Eliminating sub expressions
c. Eliminating loop invariant variables
d. Eliminating induction variables

228. In labeling algorithm, that n is a binary node and its children have l_1 and l_2, LABEL (n) if $l_1 \neq l_2$ is
a. Max (l_1, l2)
b. l_2+1
c. $l_1 + l_1$
d. l_1 +1

229. The following instruction immediately following an unconditional jump may be removed is

a. Labeled instruction
b. Break
c. Main
d. Unlabelled

230. The ordering and multiple match problem can be resolved by using
a. Tree matching patterns
b. Dynamic programming pattern
c. Tree matching patterns in conjunction with dynamic programming pattern
d. Pattern recognition

231. The looping problem can be resolved using
a. State splitting
b. Forking
c. Union
d. Joining