

PPL

1. Use of recursion :->Enhances logical clarity
2. . If the postfix equivalent of the statement if c then x else y is cxy#, then the postfix form amn+mn-ab-# ba-# :->Is equivalent to, if a then m+n then if m-n else a-b else b-a.
3. Extensive non deterministic mechanisms which cover statements as well are provided by:->Algol 68
4. From the point of view of clarity and efficiency, \_\_\_\_\_ is preferable:->Application-order evaluation
5. val is a well known:->Data Flow language
6. A recursive function f, is defined as follows:  $F(n)=2$ , if  $n=0$   $=m$ , if  $n=1$   $=2 * F(n-1) + 4 * F(n-2)$ , if  $n \geq 2$  If the value of  $F(4)$  is 88, then the value of m is :->1
7. Which language is case-sensitive?:->C
8. \_\_\_\_\_ is used in PL/I language as a terminator and in pascal language as a separator:->Semi-colon
9. FORTRAN does not permit recursion because :->It uses static allocation for storing variables
10. Binding (of an identifier to a value) can occur while:->Invoking a sub-program
11. Which language has no For loop: :->Modula-1
12. The language which prohibit changes to the loop index within the body of an Enumeration-controlled loop is: :->Algol 68
13. Loops come in \_\_\_\_\_ principal varieties: :->Two
14. A \_\_\_\_\_ loop is executed until some Boolean condition changes Value: :->logically controlled
15. Which language introduced a mid test or one-and-a-half loop that allows a terminating condition to be tested as many items as desired within the loop:->Modula 1
16. \_\_\_\_\_ provides a single loop construct that subsumes the properties of more Modern enumeration and logically controlled loops: :->Algol 60
17. Modern For loops reflect the impact of both \_\_\_\_\_ and \_\_\_\_\_ Challenges:->Semantic, implementation.
18. The lambda keyword is used to introduce \_\_\_\_\_ :->Function
19. A function in which additional computation never follows a recursive call: the return value is simply what ever the recursive call returns is called:->Tail-Recursive function
20. Which language do not permit recursion:->Fortran 77
21. Continuation support in scheme takes the form of a general purpose function Called \_\_\_\_\_ abbreviated call/cc :->Call\_with\_current\_continuation.
22. In the event of a nonlocal goto, the language implementation must guarantee to Repair the run time stack of subroutine call information. This repair operation is Known as \_\_\_\_\_ :->Unwinding
23. Gotos were used fairly often in \_\_\_\_\_ to terminate the current subroutines: :->Pascal
24. Short circuiting changes the \_\_\_\_\_ of Boolean expressions: :->Semantics
25. In the code below CASE \_\_\_\_\_ (\*potentially complicated expression\*) OF | : clause-A | 2, 7 : clause-B | 3...5 : clause-C | 10 : clause-D ELSE clause-E END The elided code fragments (clause-A, clause-B, etc) after the colons and the ELSE are called \_\_\_\_\_ of CASE statements. :->arms
26. Which language does not use short-circuit evaluation: :->Pascal
27. In which languages does both the THEN clause and the ELSE clause are defined To contain a single statement: :->Algol 60, Pascal.
28. A \_\_\_\_\_ loop is executed once for every value in a given Finite set :->enumeration-controlled.
29. In algol 60, a switch is essentially an array of \_\_\_\_\_ :->labels
30. Modern case statements are a descendent of the computed \_\_\_\_\_ statement of Fortran and the construct of Algol 60. :->goto, switch
31. Clu and ada use \_\_\_\_\_ and \_\_\_\_\_ respectively, For "not equal" :-> $\neq$  and  $\neq$
32. The following code is valid in; Begin a:= if b < c then d else e; a = begin f(b); g(c) end; g(d); 2+3 End :->algol 68
33. The first language to make orthogonality a principal design goal is :->a
34. Language that implements reference model is :->Haskell



35. Many of structured control-flow constructs familiar to modern programmers Were pioneered by \_\_\_\_\_ :->**algol 60**
36. The language designers debated hotly the merits and evils of gotos Throughout the late \_\_\_\_\_ and much of the \_\_\_\_\_ :->**1960s and 1970s**
37. Early versions of Fortran mimicked the low-level approach by relying Heavily on \_\_\_\_\_ statements for most nonprocedural control flow; :->**goto**
38. \_\_\_\_\_ provide mechanisms for both multilevel returns and exceptions, but this Dual support is relatively rare: :->**Lisp and Ruby**
39. Case statements were adopted in limited form by \_\_\_\_\_ and more completely by \_\_\_\_\_ :->**Algol 68 , Modula 1**
40. The \_\_\_\_\_ statement was introduced by wirth and Hoare in the algol W [W6HH]: :->**case**
41. In C++, a+b is short for: :->**a. operator + (b)**
42. In Ada, a+b is short for: :->**"+" (a, b)**
43. In Algol-family languages, function calls consists of the syntax: :->**my\_func(A,B,C)**
44. A potentially complex collection of control constructs encapsulated in a way that Allows it to be treated as a single unit, often subject to parameterization is called as \_\_\_\_\_ :->**Procedural abstraction**
45. The notation that occurs in small talk and also occasionally in Algol-family Language is called; :->**Multiword infix notation**
46. Lisp uses \_\_\_\_\_ notation for all functions but places the function name \_\_\_\_\_ The parentheses: :->**prefix, inside**
47. Most imperative languages use \_\_\_\_\_ function for binary operators and \_\_\_\_\_ function for unary operators. :->**infix , postfix**
48. \_\_\_\_\_ in a purely functional language are said to be Referentially transparent. :->**expressions**
49. Which language syntax does not allow unparenthesized form: :->**ada**
50. In FORTRAN, which uses \*\* for exponentiation, how should we parse  $a+b*c**d**e/f$ ? :->**a+ ((b\*(c\*\* (d\*\*e)))/f)**
51. Semantic rules attached to the productions of a \_\_\_\_\_ grammar are used to define the attribute flow of a parse tree. :->**context free**
52. Semantic rules attached to the productions of a \_\_\_\_\_ grammar can be used to define the attribute flow of a syntax tree: :->**tree**
53. A \_\_\_\_\_ grammar is meant to define (or generate) the trees themselves: :->**tree**
54. A \_\_\_\_\_ grammar is meant to define (generate) a language composed of strings of tokens, where each string is the fringe (yield) of a parse tree. :->**context free**
55. In an \_\_\_\_\_, action routines can be embedded at arbitrary points in a production's right-hand side: :->**LL parsers**
56. \_\_\_\_\_ grammar associates attributes with each symbol in a \_\_\_\_\_ grammar, and attribute rules with each production: :->**attribute, context-free**
57. . In \_\_\_\_\_, the error messages are buffered and printed in the program order at the end of compilation: :->**multipass compiler**
58. Space for attributes in a \_\_\_\_\_ compiler can be allocated automatically, or managed explicitly by the writer of action routines: :->**top-down**
59. Space for attributes in a \_\_\_\_\_ compiler is naturally allocated in parallel with the parse stack: :->**bottom-up**
60. In an \_\_\_\_\_, action routines must follow the production's left corner: :->**LR parsers**
61. \_\_\_\_\_ is based on formal logic and was devised as tool for proving the correctness of programs: :->**axomatic semantics**
62. \_\_\_\_\_ is a method of describing the meaning of language constructs in terms of their effects on an ideal machine: :->**operational semantics**
63. L-attributed grammars are a \_\_\_\_\_ of S-attributed grammars. :->**proper superset**
64. In L-attributed grammar, its attributes can be evaluated by visiting the nodes of the parse tree in a single \_\_\_\_\_, \_\_\_\_\_ traversal. :->**left to right, depth-first**
65. For a \_\_\_\_\_ parser with an \_\_\_\_\_ grammar, the obvious approach is to maintain an attribute stack that directly mirrors the parse stack. :->**bottom-up, S-attributed**



66. A translator that interleaves semantic analysis and code generation with parsing is said to be \_\_\_\_ :->**one-pass compiler**
67. In \_\_\_\_, mathematical objects are used to represent the meanings of language constructs. :->**denotational semantics**
68. In a \_\_\_\_ grammar we should need to pass symbol table information into and out of productions through inherited and synthesized attributes. :->**pure attribute**
69. In both families of parsers, it is common for some of the contextual information for action routines to be kept in \_\_\_\_ :->**global variables**
70. For a \_\_\_\_ parser with an \_\_\_\_ grammar, there are two principal options for space management. One is automatic and other is lower space overhead. :->**top-down, L-attributed**
71. An attribute grammar is \_\_\_\_ if it never leads to a parse tree in which there are cycles in the attribute flow graph. :->**noncircular**
72. \_\_\_\_ attributes pass semantic information down and across a tree. :->**Inherited attributes**
73. \_\_\_\_ attributes are synthesized attributes of leaf nodes whose values are determined outside the parse tree. :->**Intrinsic attributes**
74. \_\_\_\_ attributes are used to pass information up a parse tree. :->**Synthesized attributes**
75. Both context free grammars and attribute grammars are \_\_\_\_ that is, they define a set of valid trees, but they don't say how to build or decorate them. :->**declarative**
76. An attribute grammar is \_\_\_\_ if no attribute, in any parse tree, ever depends (transitively) on itself. :->**noncircular**
77. An attribute grammar is .....if attributes are guaranteed to converge to a unique value. :->**circular**
78. Better performance for noncircular grammars, may be achieved by \_\_\_\_ that tailors the evaluation order to the structure of a given parse tree. :->**dynamic scheme**
79. \_\_\_\_ fastest translation schemes, which are based on an analysis of the structure of the attribute grammar itself, and then applied mechanically to any tree arising from the grammar. :->**static scheme**
80. An algorithm that decorates parse trees by invoking the rules of an attribute grammar in an order that respects the tree's attribute flow is called a \_\_\_\_ :->**translation scheme**
81. Compile-time algorithms that predict run-time behavior are known as \_\_\_\_ analysis. :->**static**
82. Runtime overhead of dynamic checks is accepted in languages like \_\_\_\_ :->**SmallTalk and Lisp**
83. Type checking is static and precise in languages like \_\_\_\_ :->**Ada, C and ML**
84. A postcondition, specified once in the header of a \_\_\_\_ subroutine, will be checked not only at the end of the subroutine's text, but at every return statement as well, automatically. :->**Euclid**
85. \_\_\_\_ analysis determines when all references to a value will be confined to a given context, allowing it to be allocated on the stack instead of the heap, or to be accessed without locks. :->**escape**
86. \_\_\_\_ analysis determines when a variable in an object-oriented language is guaranteed to have a certain subtype, so that its methods can be called without dynamic dispatch. :->**subtype**
87. \_\_\_\_ analysis determines when values can be safely cached in registers, computed "out of order," or accessed by concurrent threads. :->**alias**
88. A compiler is said to be \_\_\_\_ if it applies optimizations only when it can guarantee that they will be both safe and effective. :->**conservative**
89. \_\_\_\_ part of compiler is a recognizer for the language the compiler translates. :->**syntax analysis**
90. Copy rules and semantic function calls are the only two kinds of permissible rules in \_\_\_\_ :->**Attribute grammars**
91. Which interface defines the boundary between front end and back end of compiler? :->**interface between semantic and intermediate code generator**
92. The annotations of a parse tree are known as \_\_\_\_ :->**attributes**
93. Compiler enforces static semantic rules at \_\_\_\_ :->**compile time**
94. Syntax of a language is described by \_\_\_\_ :->**Context-free grammar**
95. \_\_\_\_ for loops are expected to be true before and after every iteration. :->**invariant**
96. \_\_\_\_ is a condition that is expected to be true at all "clean points" of a given body of code. :->**invariant**
97. \_\_\_\_ is a statement that a specified condition is expected to be true when execution reaches a certain point in the code. :->**assertion**
98. In which programming language, a programmer can specify an invariant on the data inside a class. :->**Eiffel**
99. \_\_\_\_ is expected to be true at the end of subroutines. :->**postcondition**



100. \_\_\_\_\_ is expected to be true at the beginning subroutines.:->**precondition**
101. Which of the following problems are iterative, rather than recursive in nature?:->**Simplex method for solving a linear programming problem** *www.Examsadda.com*
102. Which can be correctly identified to be pascal tokens without look-ahead scanning :->=
103. In which cases, is it possible to obtain different results for call-by-reference and call-by-name parameters passing? :->**Passing an expression as a parameter**
104. \_\_\_\_\_ evaluates an arithmetic expression, the same way as a calculator :->**APL**
105. At which time we can resolve intermediate references:->**Link time**
106. The principle that a function can always be replaced by its value without changing the meaning is called. :->**Referential transparency**
107. Which is not dangling reference? :->**Accessing a variable that is declared and initialized**
108. The default binding in the language with static scoping is :->**Deep Binding**
109. The early binding of referencing environment is known as :->**Deep Binding**
110. The default binding in the languages with dynamic scoping is :->**Shallow Binding**
111. The binding that is generally the default in languages with static scoping is :->**deep binding**
112. The late binding of referencing environment of a subroutine that has been passed as a parameter is known as :->**shallow binding**
113. \_\_\_\_\_ rules specify that the referencing environment depends on the lexical nesting of program blocks in which names are declared :->**Static scope rules**
114. Nested subroutines are scopes in most Algol family languages:->**Open**
115. FORTRAN 95 allows upto \_\_\_\_\_ characters in its name :->**31**
116. Language designed to support large programs must provide:->**Separate Compilation**
117. The process by which a compiler automatically converts a value of one type into a value of one type into a value of another type when that second type is required by the surrounding context :->**Coercion**
118. A static chain is a chain of static links that connect certain activation record instances in the \_\_\_\_\_ :->**Stack**
119. Static scoping is a central feature of the \_\_\_\_\_ language:->**Algol 60**
120. \_\_\_\_\_ are useful as aids to readability and program reliability :->**Named constants**
121. The bindings between names and objects can be determined at compile time by examining the text of program, without consideration of flow of control at run time, in the languages with \_\_\_\_\_ scoping:->**Static**
122. Which of the language is statically scoped:->**C**
123. The heap is divided into \_\_\_\_\_ one for each standard size:->**Pools**
124. Intermediate values produced in complex calculations are:->**Temporaries**
125. How to divide the effort among programmers in such a way that work can proceed on multiple \*\*\*\*\* simultaneously is a major challenge in :->**Construction of any large body of software**
126. A name-to-object binding that is hidden by a nested declaration of the same name is said to have a \_\_\_\_\_ in its scope:->**Hole**
127. The ability to nest subroutines inside each other is not a feature in the \_\_\_\_\_ language:->**Smalltalk**
128. In which language we do **not** find modules:->**Algol 60**
129. A module does not allows a collection of objects-subroutines, variables, types and so on to be encapsulated in such a way that :->**Objects outside are visible to each other**
130. Modules in clu are called as :->**Clusters**
131. The programming language that supports both functional and imperative programming is \_\_\_\_\_ :->**ML** *www.Examsadda.com*
132. In java script and PHP, the binding of a variable to a type is:->**Dynamic**
133. In perl any name that begins with % is a \_\_\_\_\_ :->**Hash structure**
134. In perl any name that begins with \$ is a \_\_\_\_\_ :->**Scalar**
135. Which language is not strongly typed :->**Fortran 95**
136. Explicit heap dynamic variables are often used for dynamic structures, such as \_\_\_\_\_ that need to grow and/or shrink during execution:->**Linked lists**
137. The storage binding of which variables are created when their declaration statements are elaborated, but whose types are statically bound:->**Stack-dynamic variables**



138. A binding to an object that is no longer live is called:->**Dangling reference**
139. \_\_\_\_\_ scoping is based on the calling sequence of subprograms, not on their spatial relationship to each other:->**Dynamic**
140. The scope of variables is not dynamic in :->**Algol 60**
141. Heap allocation is required for languages that:->**Support dynamic data structures**
142. The time at which any implementation decision is made is called as:->**Binding time**
143. Aliasing is a situation where:->**Two commands with different names share the same code**
144. The period of time between an allocation and its subsequent disposal is called:->**Life time**
145. A call to a library sub program is bound to the sub program code at:->**Link time**
146. The time at which the programmers, choose algorithms, data structures and names is :->**Program writing time**
147. For which of the following application will you prefer a co-routine to a subroutine? :->**Simulation of multi-processing**
148. Programming languages designed since the \_\_\_\_\_ require explicit declarations of all variables:->**Mid-1960's**
149. A binding is dynamic if it first occurs during \_\_\_\_\_ or can change in course of program execution :->**run time**
150. A binding is static if it first occurs before \_\_\_\_\_ and remains unchanged throughout program execution:->**run time**
151. Which of the following is true about reliability? :->**Readability and writability influence reliability**
152. Languages designed around the prevalent computer architecture, called the von Neumann architecture are called as \_\_\_\_\_ :->**Imperative Languages**
153. Which of the following are von-Neuman languages? :->**C, Ada, Fortran**
154. \_\_\_\_\_ is the purest of the object-oriented languages. :->**Smalltalk**
155. Pure LISP has only two kinds of data structures: atoms and \_\_\_\_\_ :->**Lists**
156. \_\_\_\_\_ language support a range type in its Switch-Case statement :->**Ada**
157. Aliasing is :->**Having two or more distinct referencing methods or names for the same memory cell is called aliasing**
158. \_\_\_\_\_ is the ability to define and then use complicated structures or operations that allow many of the details to be ignored:->**abstraction**
159. IPL stands for \_\_\_\_\_ :->**Information Processing Language**
160. Internally lists are usually stored as \_\_\_\_\_ :->**Single-linked list structure**
161. Regular expressions and context-free grammars are \_\_\_\_\_ :->**Language generator**
162. Any set of strings that can be defined if we add recursion is called a \_\_\_\_\_ :->**context-free language**
163. A parser is a concrete realization of a \_\_\_\_\_ :->**Deterministic Push Down Automaton**
164. A scanner is a \_\_\_\_\_ that recognizes the tokens of a programming language:->**Deterministic Finite Automaton**
165. Which scripting language is developed by Netscape for use in both Web server and browsers?:->**Java Script**
166. Which language is used widely for Artificial Intelligence Application? :->**LISP**
167. Scanners and parsers are \_\_\_\_\_ :->**Language recognizers**
168. Which was the first high-level language developed for business purpose? :->**COBOL**
169. Which was the first language for scientific applications? :->**FORTTRAN**
170. Which language is developed to produce business reports? :->**RPG**
171. The sentences of the language are generated through a sequence of applications of the rules, beginning with a special nonterminal of the grammar called the \_\_\_\_\_ :->**Start Symbol**
172. A BNF description or grammar, is simply a collection of \_\_\_\_\_ :->**Rules**
173. In BNF grammar, the lexemes and tokens of the rules are called:->**Terminals**
174. The abstractions in a BNF description, or grammar, are called :->**Non terminals**
175. \_\_\_\_\_ is a language that is used to describe another language:->**Meta Language**
176. One feature of grammars is, they describe the hierarchial syntactic structure of the sentences of the languages. These hierarchial structures are called \_\_\_\_\_ :->**Parse Trees**
177. Each of the strings in the derivation is called \_\_\_\_\_ :->**Sentential form**
178. The brackets, braces and parentheses in the EBNF are \_\_\_\_\_ :->**metasymbols**



179. When a grammar rule has its LHS appearing at the right end of the RHS, the rule is said to be \_\_\_\_\_  
:->right recursive *www.Examsadda.COM*
180. When a grammar rule has its LHS also appearing at the beginning of its RHS, the rule is said to be \_\_\_\_\_  
:->left recursive
181. What are functional programming languages? :->Lisp/Scheme, ML, Haskell
182. Find out the declarative programming languages from the following:->Lisp/Scheme, Prolog
183. B Programming Language is designed by :->Ken Thompson
184. Linux, the leading open source operating system is written in :->C
185. The internal codes are called \_\_\_\_\_ :->tokens
186. The character groupings are called \_\_\_\_\_ :->lexemes
187. The \_\_\_\_\_ phase of the compiler collects characters into logical groupings and assigns internal codes to groupings according to their structure.:->Lexical analyser
188. Which of the following is a device that can be used to generate the sentences of a language? :->Language generator
189. A sentence generation is called a \_\_\_\_\_ :->derivation
190. Tokens are usually coded as \_\_\_\_\_ values for readability of lexical and syntax analyzers. :->integer
191. Translating high level language instructions to assembly language instructions or machine language instructions is the job of the system program known as \_\_\_\_\_ :->compiler
192. Translating mnemonics to machine language instructions is the job of the system program known as \_\_\_\_\_ :->assembler
193. Assembly Language instructions are designed with \_\_\_\_\_ correspondence between mnemonics and machine language instructions. :->one to one
194. What is MIPS R4000? :->Microprocessor
195. Prolog is good for :->reasoning about logical relationships about data
196. C is good for :->low-level systems programming
197. Lisp is good for :->manipulating symbolic data and complex data structures
198. Logo is popular among :->Novice users
199. In a formal mathematical sense a language is :->Turing equivalent
200. Snobol and Icon are good for :->manipulating character strings

*www.Examsadda.COM*