|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **P.V.P Siddhartha Institute of Technology(Autonomous)** | | | | | | | | | | | | | | | | **Signature of Invigilator with date:** | | **Marks Obtained:** | |
| **Department of Computer Science and Engineering** | | | | | | | | | | | | | | | |
| **Course: B. Tech** | | **Year: III** | | | **Semester: II** | | | **Objective: I** | | | | | | | |
| **Regulation:PVP20** | | **Maximum Marks:10Marks** | | | | | | | | | **Session: F. N** | | | | |
| **A.Y:2023-24** | | **Date:29-01-2024** | | | | | | | **Duration: 20 min** | | | | | | |
| **Subject Name: Compiler Design** | | | | | | | | | | | | | | | | | | | | |
| **Registered Number:** | | | | | | | | **Name:** | | | | | | | | | | | | |
| **Answer all the Questions. Each Question carries ½ Mark 20×½ M =10M** | | | | | | | | | | | | | | | | | | | | |
| **S. No** | **Question** | | | | | | | | | | | | | | | | **CO** | **Level** | **Answer** |
| 1. | **Compiler takes\_\_\_\_ as input and interpreter takes\_\_\_\_\_\_\_ as input respectively.** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| 1. Single instruction, Entire program. | | | 1. Entire program, Single instruction | | | | 1. Entire program | | | | | | | 1. Single instruction | |
| 2. | **Which of the following is the third phase of Compiler?** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. Lexical Analysis | | | | | | | b. Syntax Analysis | | | | | | | | |
| c. Semantic Analysis | | | | | | | d. Code Generation | | | | | | | | |
| 3. | **Which of the following are two stages of a compiler design** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| 1. Analysis, Semantic | | | | | | | 1. Analysis, Synthesis | | | | | | | | |
| 1. Assembly, Syntactical | | | | | | | 1. Assembly, Semantic | | | | | | | | |
| 4. | **Which of the following phase produces parse tree as output?** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. Lexical Analysis | | | | | | | b. Syntax Analysis | | | | | | | | |
| c. Code optimization | | | | | | | d. Code Generation | | | | | | | | |
| 5. | **LEX tool used to generate \_\_\_\_\_\_\_\_\_\_\_\_.** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. lexical  Analyzer | | | b. Syntax  Analyzer | | | | c. Semantic  Analyzer | | | | | | | d. Code Analyzer | |
| 6. | **Which of the following values are associated for each token** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| 1. attribute values | | | 1. Token name | | | | | 1. Both a and b | | | | | | 1. None | |
| 7. | **The regular expression a\* generates \_\_\_\_.** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. a | | | b. aa | | | | c. € | | | | | | | d. All | |
| 8. | **Syntax analyzer is also called as** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. Scanner | | | b. Analyzer | | | | c. Parser | | | | | | | d. None | |
| 9. | **Type checking will be done in which of the following phase.** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| 1. Lexical Analysis | | | 1. Syntax Analysis | | | | 1. Semantic Analysis | | | | | | | 1. Code Generation | |
| 10. | **Find number of tokens in the following code: int x==y;** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. 6 | | | b. 5 | | | | c. 4 | | | | | | | d. 3 | |
| 11. | **Lexical Analyzer groups the characters into meaningful sequences called \_\_.** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. Pass | | | b. Lexeme | | | | c. LEX | | | | | | | d. Phase. | |
| 12. | **Which of the following is Pattern for Identifiers** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. [a-zA-Z][a-zA-Z0-9]\* | | | b. [0-9]\* | | | c. [a-z]\* | | | | | d. if|else|switch | | | | |
| 13. | **Consider the following grammar S ->A# A ->+/ € then FOLLOW(A) =** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| 1. + | | | | | | 1. \* | | | | | | | | | |
| 1. B | | | | | | 1. # | | | | | | | | | |
| 14. | **Consider the following grammar S ->Ab A ->+/ € then FIRST(S) =\_\_\_\_.** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. **{+}** | | | b. {+, b} | | | | c. {$} | | | | | | d. {A} | | |
| 15. | **r?=r/€ here ? represents** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| 1. One or more instance | | | 1. Zero or more instance | | | | 1. Zero or one instance | | | | | | | 1. None | |
| 16. | **A LEX program has \_\_\_\_\_\_\_\_ number of sections** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. 1 | | | b. 3 | | | | c. 2 | | | | | | | d. 4 | |
| 17. | **In Parse tree, leaf nodes are called?** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. Terminals | | | b. Non terminals | | | | c. Sub terminals | | | | | | | d. Half terminals | |
| 18. | **A bottom up parser generates** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. Right Most Derivation | | | | | | | b. Right Most derivation in reverse | | | | | | | | |
| c. Leftmost derivation | | | | | | | d. Leftmost derivation in reverse | | | | | | | | |
| 19. | **Shift reduce parsing belongs to class of** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a. Bottom up parsing | | | | | b. Top down parsing | | | | | | | | | | |
| c. Recursive parsing | | | | | d. Predictive parsing | | | | | | | | | | |
| 20. | **The grammar E -> aS/aB/b is \_\_\_\_\_\_** | | | | | | | | | | | | | | | | **CO1** | **L2** |  |
| a . Left recursive | | b. Left factored | | | | | | | c. right recursive | | | d. None | | | |