Symbol Table Requirements:

1. Structural Flexibility: The symbol table should be able to store all necessary information about identifiers based on how they are used in the program.
2. Fast Lookup/Search: Searching for entries in the symbol table should be quick, so that programs can efficiently access information about identifiers.
3. Efficient Space Utilization: The symbol table should be able to adjust its size dynamically to use memory efficiently.
4. Handling Language Characteristics: It should be capable of handling language features like scoping and implicit declaration, which affect how identifiers are treated.

Demerits of Uniform Structure:

1. Limited Name Length: If the symbol table has a fixed limit on the length of identifier names it can store, longer names won’t fit, causing problems.
2. Waste of Space: If the symbol table reserves a fixed amount of space for each entry, shorter names will leave unused space, which is inefficient.

Related Posts:

1. What are the types of passes in compiler ?
2. Discuss the role of compiler writing tools. Describe various compiler writing tools.
3. What do you mean by regular expression ? Write the formal recursive definition of a regular expression.
4. How does finite automata useful for lexical analysis ?
5. Explain the implementation of lexical analyzer.
6. Write short notes on lexical analyzer generator.
7. Explain the automatic generation of lexical analyzer.

8. Explain the term token, lexeme and pattern.
9. What are the various LEX actions that are used in LEX programming ?
10. Describe grammar.
11. Explain formal grammar and its application to syntax analyzer.
12. Define parse tree. What are the conditions for constructing a parse tree from a CFG ?
13. Describe the capabilities of CFG.
14. What is parser ? Write the role of parser. What are the most popular parsing techniques ? OR Explain about basic parsing techniques. What is top-down parsing ? Explain in detail.
15. What are the common conflicts that can be encountered in shift-reduce parser ?
16. Differentiate between top-down and bottom-up parser.Under which conditions predictive parsing can be constructed for a grammar ?
17. Differentiate between recursive descent parsing and predictive parsing.
18. What is the difference between S-attributed and L-attributed definitions ?
19. What is intermediate code generation and discuss benefits of intermediate code ?
20. Define parse tree. Why parse tree construction is only possible for CFG ?
21. Discuss symbol table with its capabilities ?