calm the mind

VIVA Q&A

B.TECH CSE

5th SEM

SNAPED

COMMUNITY

Compiler Design

“ MINDSET FIRST, ACTION SECOND ”

At SnapED codeCampus, we believe in accessible, affordable, and high-quality education for all. SnapED will help you become industry-ready with skill-boosting courses and guides, including DSA with Java and C++, Web Development, Python, Machine Learning, and more. Dive into expertly curated resources, prepare smarter with important notes and questions, and achieve your academic goals seamlessly.

Explore more at SnapED codeCampus:

- Notes

- Important Questions

- Previous Year Questions (PYQs)

- Video Lectures

- Viva Ques

- skill-boosting courses and guides.

Stay Connected with Us:

YouTube: SnapED codeCampus

LinkedIn: SnapED codeCampus

WhatsApp Community:

1st Community

2nd Community

"Your success story begins here."

Viva Questions and Answers

---

1. Practice of LEX/YACC

1. What is the role of Lex in compiler design?

Answer: Lex is a tool used to generate a lexical analyzer, which converts a sequence of characters into tokens for parsing.

2. What is YACC?

Answer: YACC (Yet Another Compiler Compiler) is a tool used to generate parsers based on context-free grammars.

3. What is a token?

Answer: A token is a sequence of characters that represents a unit of syntax, such as keywords, identifiers, operators, or literals.

4. What is the output of Lex?

Answer: Lex generates a C program (lex.yy.c) that implements the lexical analyzer.

5. How does Lex interact with YACC?

Answer: Lex generates tokens, which are passed to the parser created by YACC to analyze syntax.

---

2. Check if a String Belongs to a Grammar

1. What is a grammar in the context of compiler design?

Answer: A grammar is a set of production rules defining the syntax of a language.

2. What is the significance of the start symbol in grammar?

Answer: The start symbol is the initial non-terminal from which all valid strings in the language are derived.

3. What are terminals and non-terminals?

Answer: Terminals are the basic symbols of the language, while non-terminals are placeholders for patterns of terminals.

4. How do you check if a string belongs to a grammar?

Answer: By deriving the string using the grammar’s production rules or by constructing a parse tree.

5. What is the difference between LL and LR grammars?

Answer: LL grammars are parsed from left to right using leftmost derivation, while LR grammars are parsed using rightmost derivation in reverse.

---

3. Check if a String Includes a Keyword

1. What is a keyword in programming languages?

Answer: A keyword is a reserved word that has a predefined meaning in the programming language.

2. How can keywords be identified during lexical analysis?

Answer: Keywords are matched against a predefined list during the lexical analysis phase.

3. What is the difference between keywords and identifiers?

Answer: Keywords have predefined meanings, whereas identifiers are names defined by the user.

4. What is the role of `strstr()` in C?

Answer: The `strstr()` function finds the first occurrence of a substring in a string.

5. Why is keyword recognition important in compilers?

Answer: It helps in categorizing tokens correctly for further syntax and semantic analysis.

---

4. Remove Left Recursion

1. What is left recursion in grammar?

Answer: Left recursion occurs when a non-terminal refers to itself as the leftmost symbol in one of its productions.

2. Why is left recursion problematic in parsing?

Answer: It causes infinite recursion in top-down parsers like LL parsers.

3. What is the difference between direct and indirect left recursion?

Answer: Direct left recursion occurs within a single production rule, while indirect left recursion involves multiple rules.

4. How can left recursion be removed?

Answer: By rewriting the grammar using auxiliary non-terminals.

5. Can left recursion exist in LR parsers?

Answer: LR parsers can handle left recursion, so it is not problematic for them.

---

5. Perform Left Factoring

1. What is left factoring?

Answer: Left factoring is a grammar transformation technique to eliminate ambiguity by factoring out common prefixes in productions.

2. Why is left factoring necessary?

Answer: It prepares a grammar for predictive parsing by reducing non-determinism.

3. What happens if left factoring is not performed?

Answer: It may lead to parsing errors or ambiguity in the grammar.

4. What is the difference between left factoring and left recursion?

Answer: Left factoring removes common prefixes, while left recursion removes leftmost self-references in rules.

5. Can left factoring introduce new non-terminals?

Answer: Yes, it often introduces auxiliary non-terminals for the remaining productions.

---

6. Implement Stack Operations

1. What are the basic operations of a stack?

Answer: Push, pop, and peek (or top).

2. What is a stack's LIFO property?

Answer: Last In, First Out means the last element added is the first to be removed.

3. What are some applications of stacks in compilers?

Answer: Parsing, expression evaluation, and function call management.

4. What happens during stack overflow?

Answer: Stack overflow occurs when more elements are pushed than the stack can hold.

5. How is a stack different from a queue?

Answer: A stack follows LIFO, while a queue follows FIFO (First In, First Out).

---

7. Find Leading of Non-terminals

1. What is leading in a grammar?

Answer: The leading symbols are the terminals that appear at the beginning of strings derived from a non-terminal.

2. How is leading computed?

Answer: By analyzing the first terminal symbols of the production rules.

3. Why is leading important?

Answer: It helps in constructing parsing tables and resolving grammar conflicts.

4. What is the difference between leading and FIRST?

Answer: Leading focuses on terminals at the start of derivations, while FIRST includes ε (epsilon) if the non-terminal can derive an empty string.

5. Can leading include non-terminals?

Answer: No, leading only includes terminal symbols.

---

8. Implement Shift-Reduce Parsing

1. What is shift-reduce parsing?

Answer: A bottom-up parsing technique that uses a stack to reduce input strings to the start symbol.

2. What are the operations in shift-reduce parsing?

Answer: Shift, reduce, accept, and error.

3. What is the role of a parsing table in shift-reduce parsing?

Answer: It guides the parser on whether to shift or reduce based on the stack and input.

4. What is a handle in shift-reduce parsing?

Answer: A substring that matches the body of a production rule and can be reduced.

5. What is the difference between shift and reduce?

Answer: Shift moves an input symbol onto the stack, while reduce replaces a handle with its corresponding non-terminal.

---

9. Find FIRST of Non-terminals

1. What does the FIRST set represent?

Answer: The set of terminals that can appear at the beginning of strings derived from a non-terminal.

2. How is FIRST calculated for a terminal?

Answer: The FIRST of a terminal is the terminal itself.

3. What happens if a non-terminal derives ε?

Answer: ε is included in the FIRST set of the non-terminal.

4. What is the significance of the FIRST set in parsing?

Answer: It helps in constructing predictive parsing tables.

5. Can the FIRST set contain non-terminals?

Answer: No, it only contains terminal symbols and ε.

---

10. Check if Grammar is Operator Precedence

1. What is operator precedence in grammar?

Answer: It defines the order in which operators are evaluated in expressions.

2. What is an operator precedence parser?

Answer: A parser that resolves conflicts by giving precedence to certain operators.

3. What is the condition for a grammar to be operator precedence?

Answer: No production should have ε or two consecutive non-terminals.

4. How is the precedence of operators determined?

Answer: By constructing a precedence table based on the grammar.

5. What is the difference between precedence and associativity?

Answer: Precedence determines the order of operations, while associativity determines the direction of evaluation.