- **Detail about Unsolvable problems:** Unsolvable problems are those that cannot be solved by any algorithm. One example is the "Halting Problem," where no general algorithm can decide if a given program will halt or continue to run indefinitely. Unsolvable problems typically involve undecidability in computational theory. These problems arise due to inherent limitations in the algorithms and computational models, such as Turing machines, indicating that certain questions will never have a definitive, algorithmic solution.

- **Discuss about Instantaneous Description in a PushDown Automata:** An Instantaneous Description (ID) of a PushDown Automata (PDA) captures the current state, the unread input, and the contents of the stack at any given moment during the computation process. It is usually represented as a tuple (q, w, s), where q is the current state, w is the remaining input string, and s represents the stack content. IDs are used to track the progression of the PDA as it processes the input, making it possible to describe how the automata behaves at any given step, particularly useful in language recognition tasks.

- **Explain the role of symbol table in compiler design:** In compiler design, the symbol table is a data structure used to store information about the identifiers (variables, functions, etc.) used in a program. It serves as a repository that tracks properties such as type, scope, and memory location of each identifier. The symbol table allows the compiler to efficiently perform semantic checks, resolve variable references, and generate correct machine code by ensuring proper variable declaration and usage.

- **What is a synthesised attribute?** A synthesized attribute is one that is computed from the attribute values of its children in a syntax tree. It is associated with a non-terminal symbol in a grammar, and the value is propagated upwards during the parse tree traversal. This is commonly used in semantic analysis, where the semantic information of a construct is derived based on its components, such as calculating the type of an expression based on the types of its sub-expressions.

- **Convert the given statement into three-address code:** The given expression `(((a-b)+(c*d))/((a-b)-(c*d)))` can be converted into three-address code as follows:

```
makefile
Copy code
t1 = a - b
t2 = c * d
t3 = t1 + t2
t4 = t1 - t2
t5 = t3 / t4
```

Here, each intermediate step breaks down the computation into individual operations with temporary variables (t1, t2, etc.), facilitating easier machine code generation and optimization.

- **Comment about unreachable code:** Unreachable code refers to portions of code that can never be executed during program runtime due to logical flaws, such as code written after a return statement. It may result from poor code structuring or logical errors in condition checks. Detecting and eliminating unreachable code is crucial for optimizing performance and preventing resource wastage, as well as improving code readability and maintainability.

- **What is a production rule?** A production rule in context-free grammar defines how a non-terminal symbol can be replaced with a sequence of terminal and/or non-terminal symbols. These rules govern how sentences in a language are derived. In formal grammar, production rules are used to generate strings belonging to the language, and they are key to constructing parsers for programming languages.

- **What is the relevance of buffer pairs and sentinels in lexical analysis?** Buffer pairs are used in lexical analyzers to store input data efficiently, allowing the system to process large amounts of text without frequent I/O operations. A sentinel is a special character placed at the end of the buffer to simplify boundary checks during scanning. The use of buffer pairs and sentinels helps in optimizing lexical analysis by reducing the need for constant boundary condition checks, improving the overall performance of the scanner.

- **For the given grammar, and an input a*b+c, construct a derivation tree:** The input expression `a*b+c` is derived using the provided grammar as follows:

```
mathematica
Copy code
E -> E + E
    -> E * E + E
    -> a * b + c
```

The derivation tree represents how the input string conforms to the grammar rules, showing how the operators and operands are hierarchically related based on precedence and associativity.

- **Describe the Regular Expression for a language Σ = {p,q}, which can contain strings whose length is divisible by 3:** The regular expression for this language consists of strings made up of symbols `p` and `q` such that the length is a multiple of 3. The regular expression would be: `((p|q)(p|q)(p|q))*`, where each group of three characters forms a valid substring. This pattern ensures that only strings with lengths divisible by 3 are accepted, as the expression is repeated in multiples of three characters.

- **Explain the functioning of a lexical analyser:** A lexical analyzer, or scanner, is the first phase of a compiler that processes the source code to divide it into tokens. These tokens are the smallest meaningful units like keywords, identifiers, and operators. The lexical analyzer reads the input character by character, groups them into tokens, and passes them to the parser for further syntactic and semantic analysis.

- **What is the relevance of code optimisation?** Code optimization is the process of transforming a program to improve its performance by reducing resource usage (such as CPU time and memory), without altering its output. This phase aims to produce more efficient code, improving execution speed and lowering power consumption. Optimization techniques include eliminating redundant computations, minimizing memory access, and reordering instructions for better hardware utilization. Efficient code leads to faster, more scalable software.