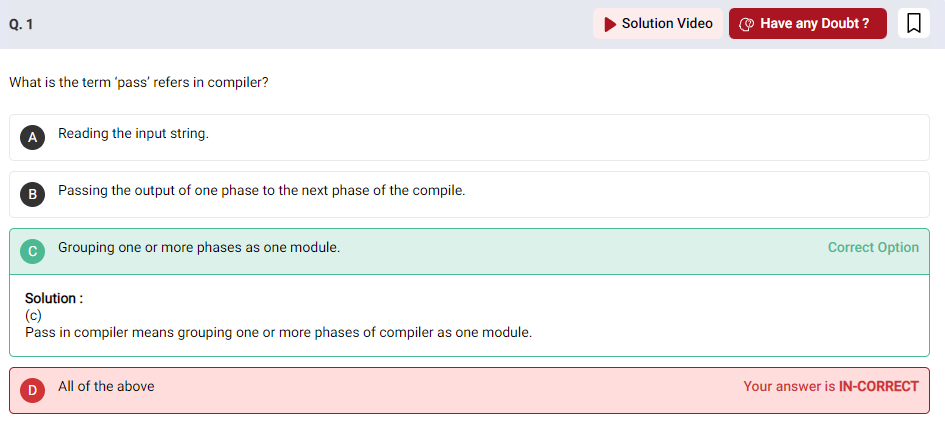
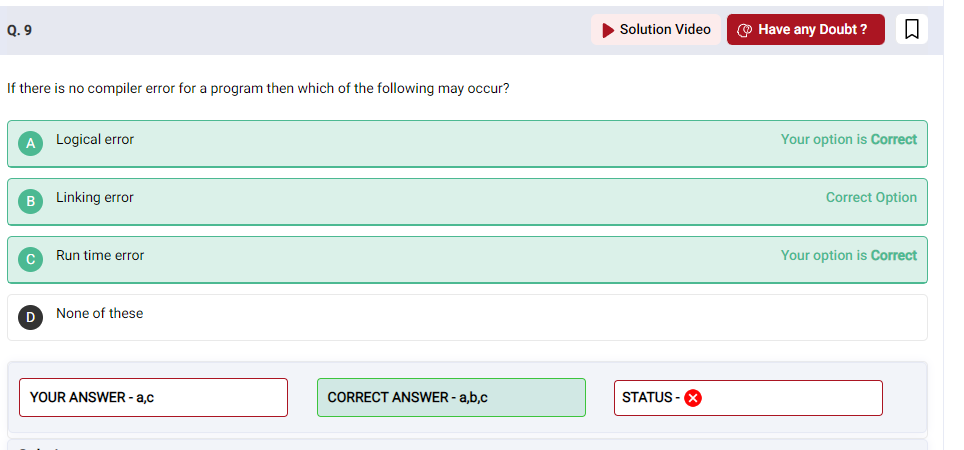
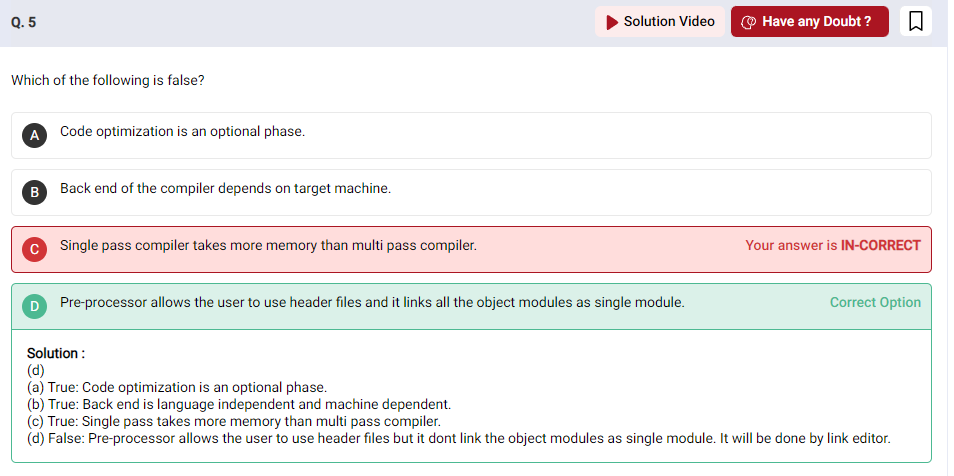
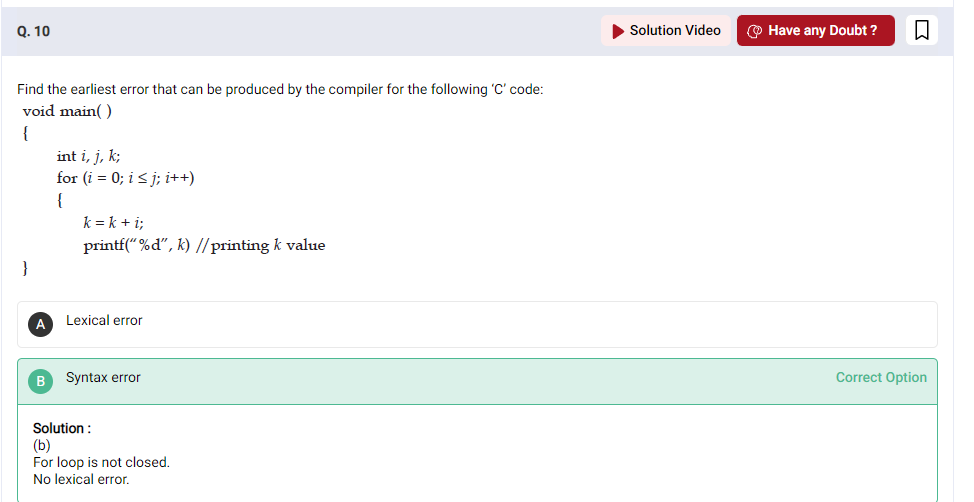
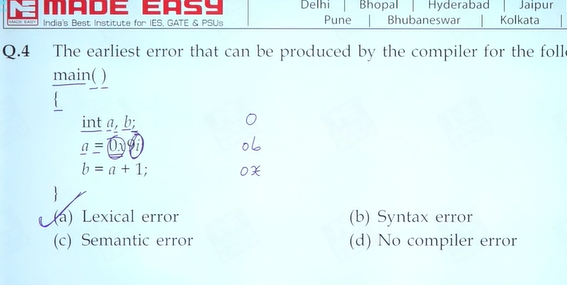
TT1:

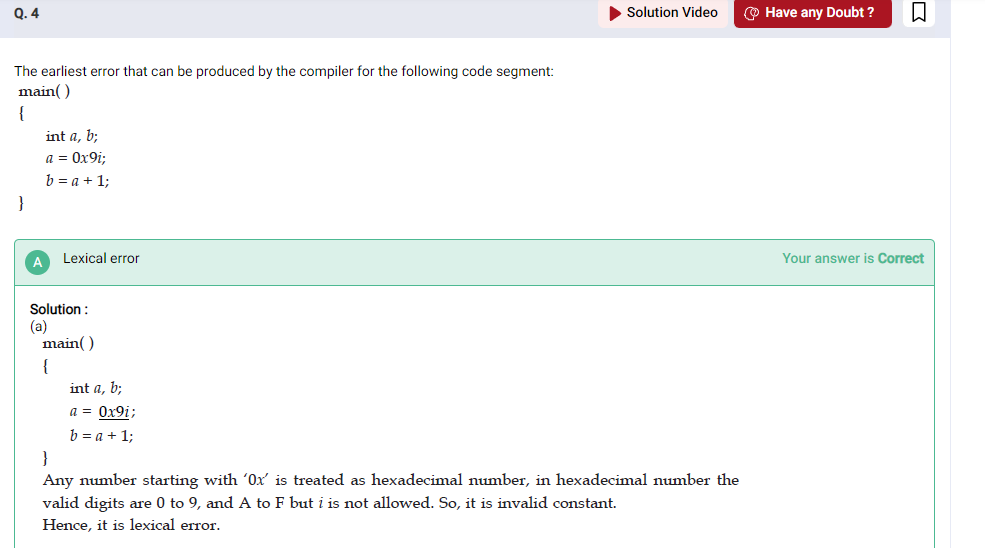


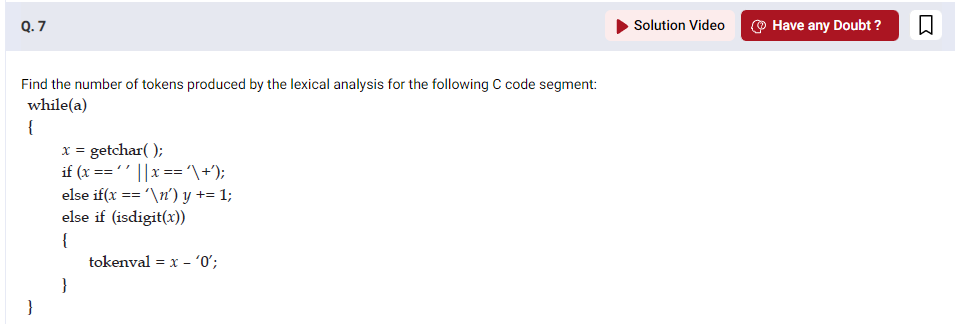


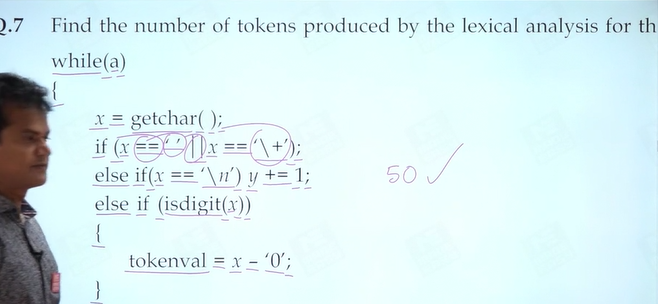


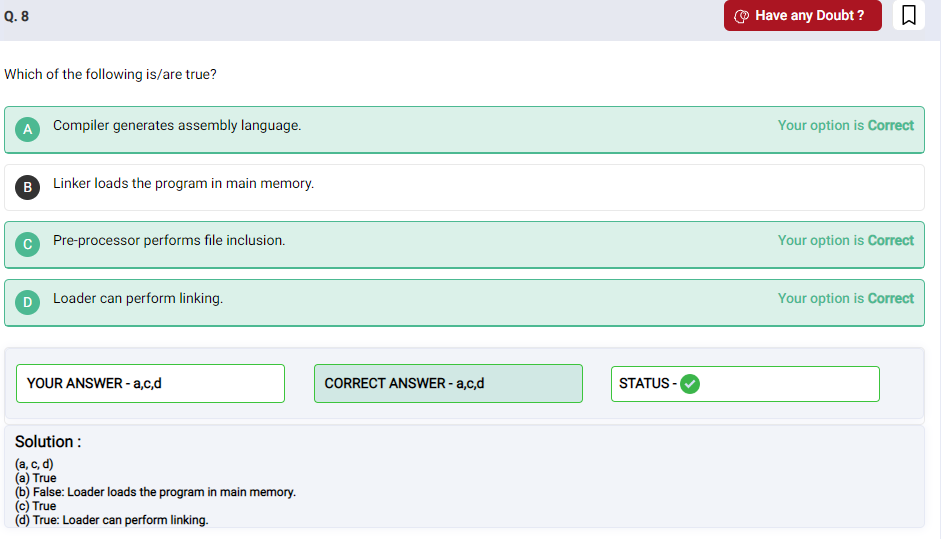


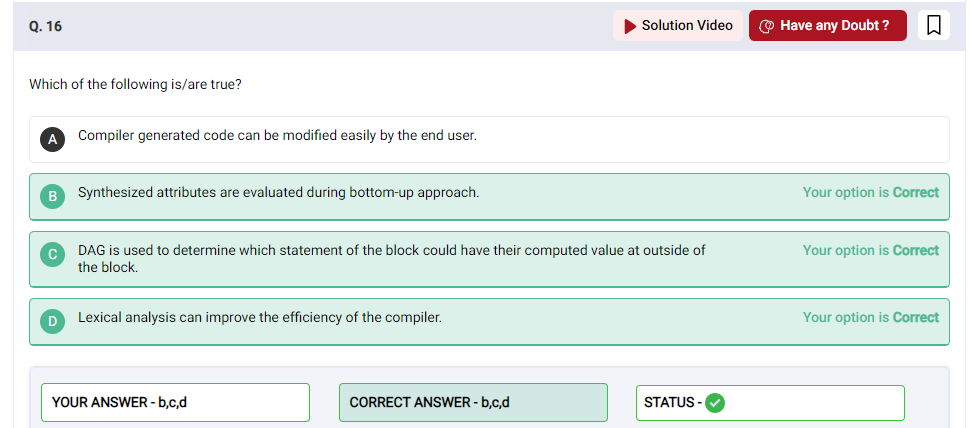


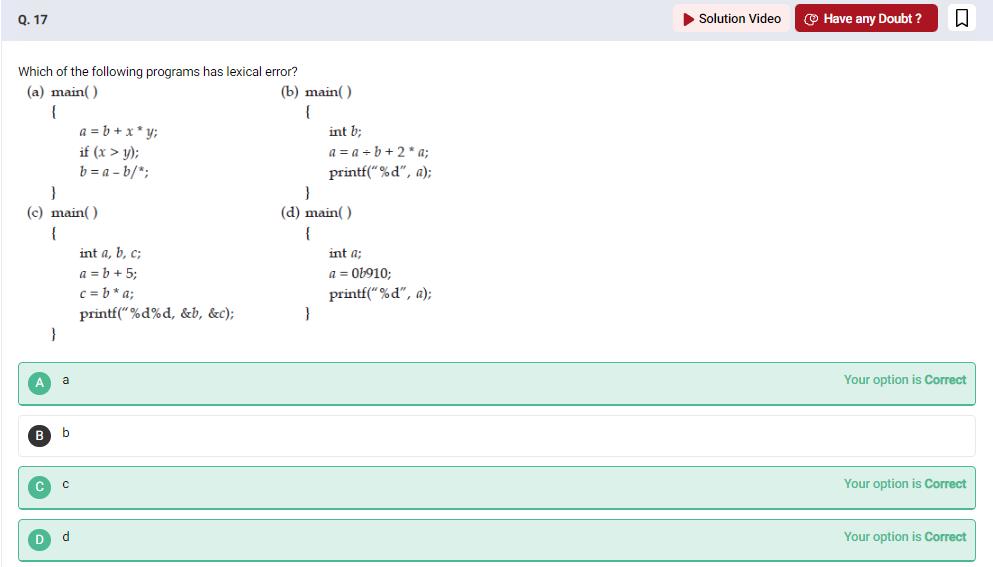




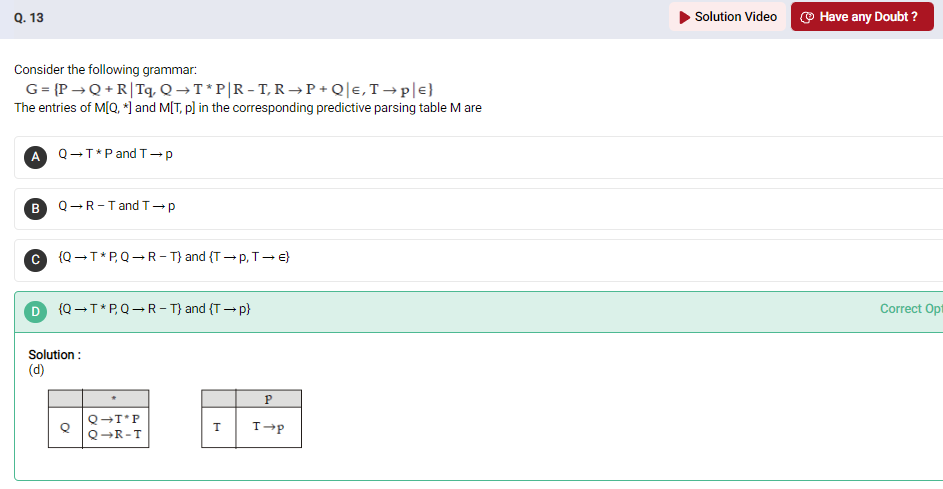


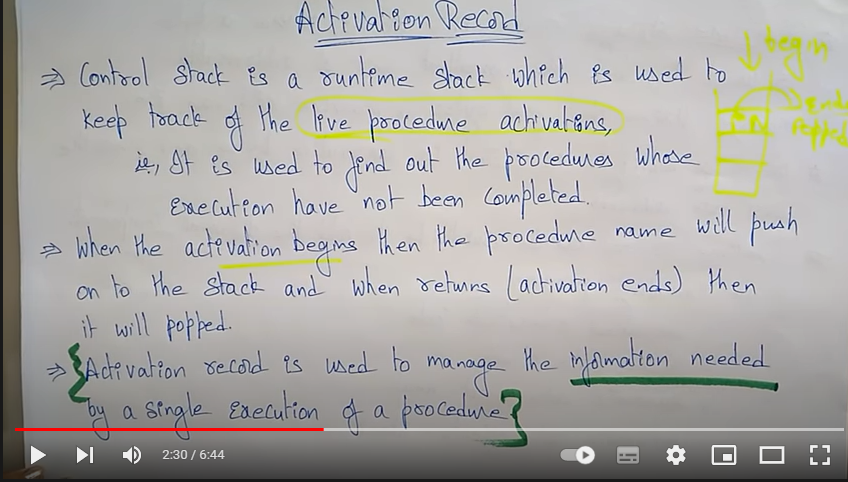


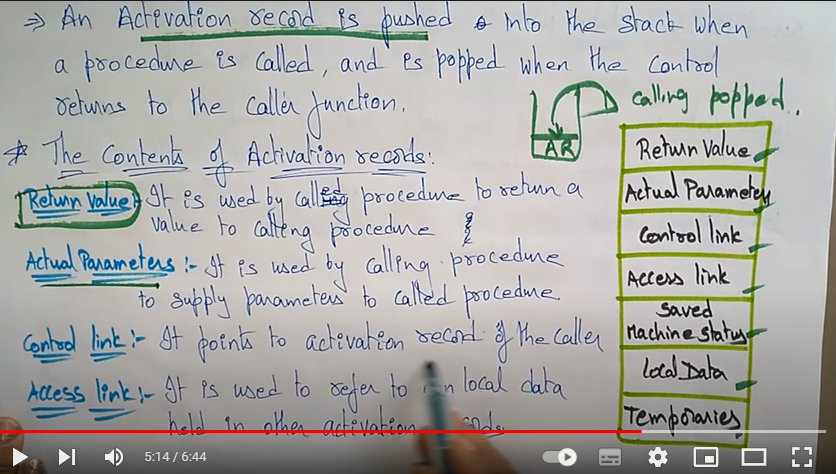


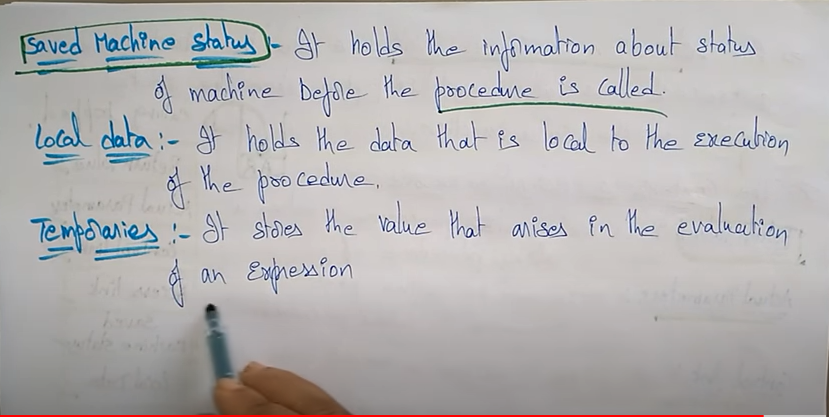


TT2:

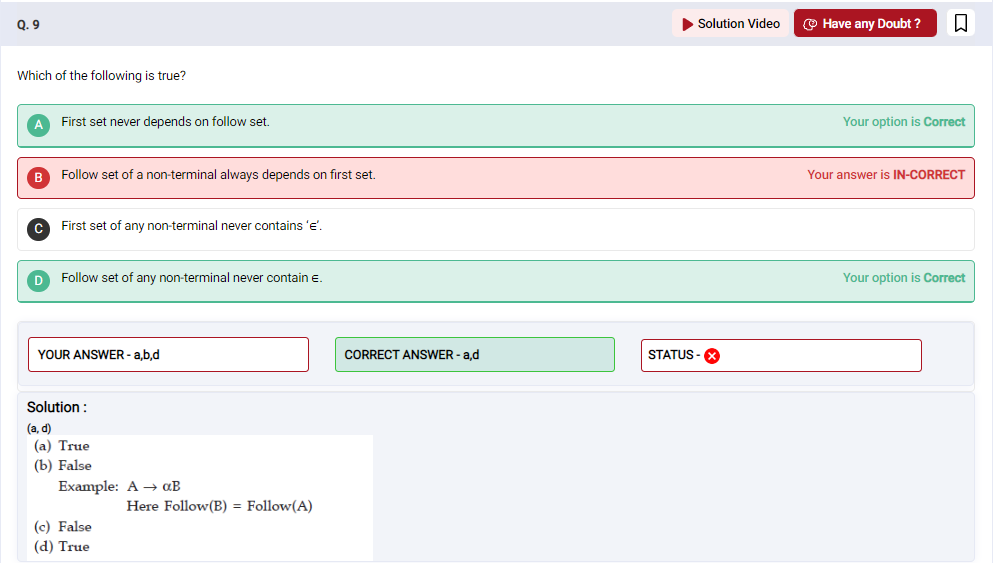


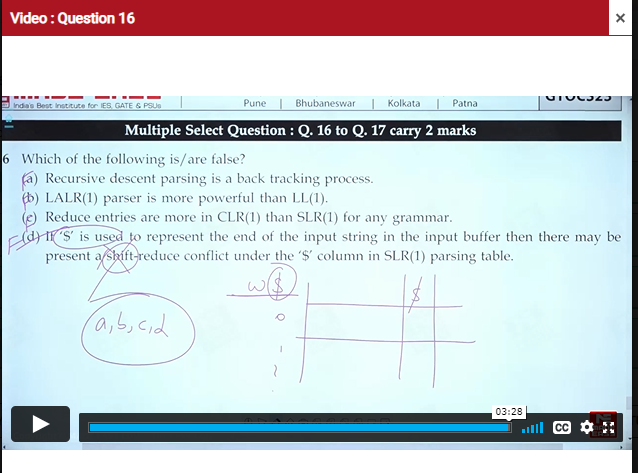


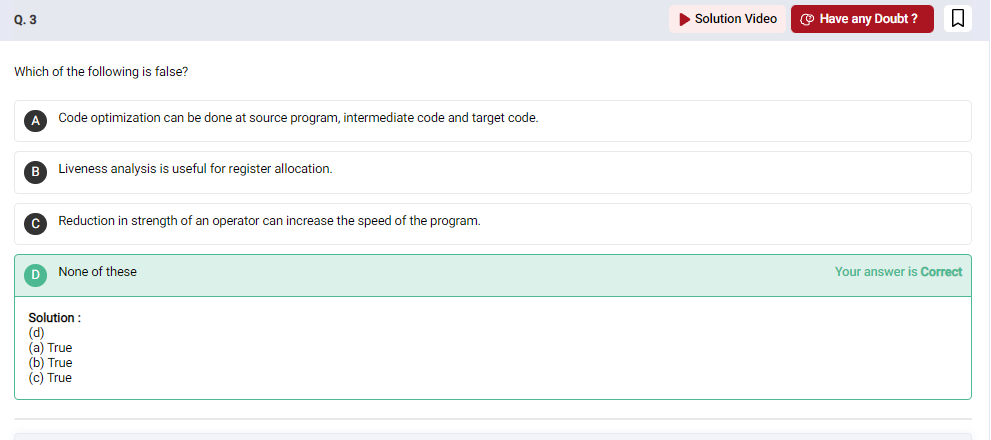


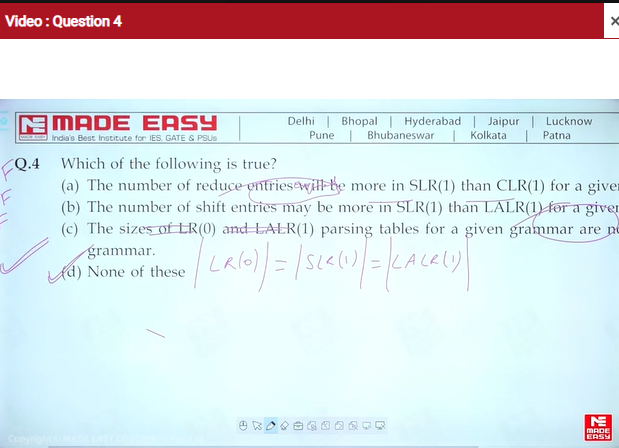


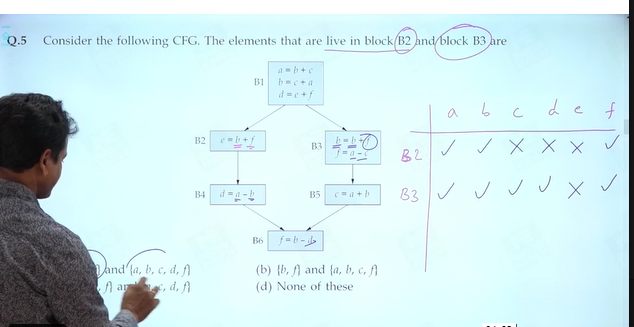




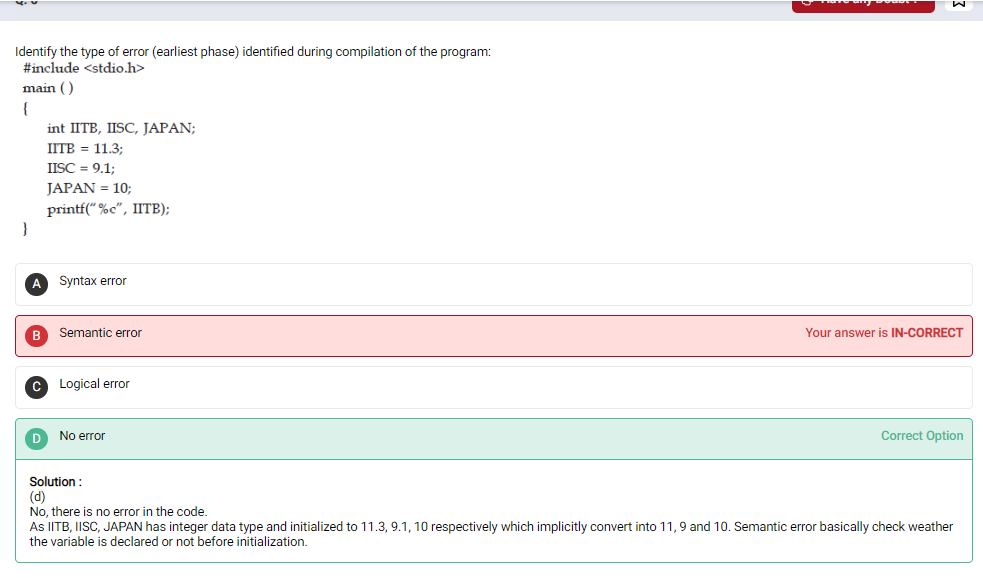
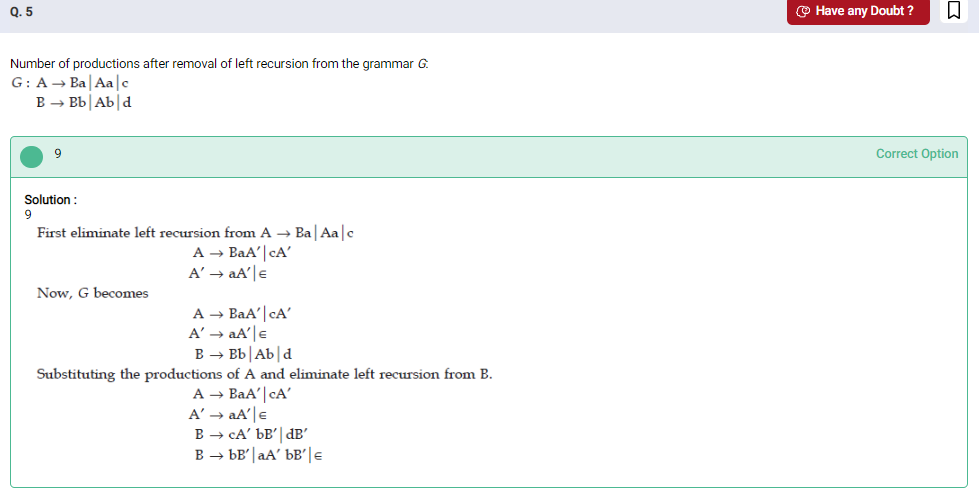








ST:

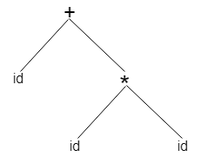
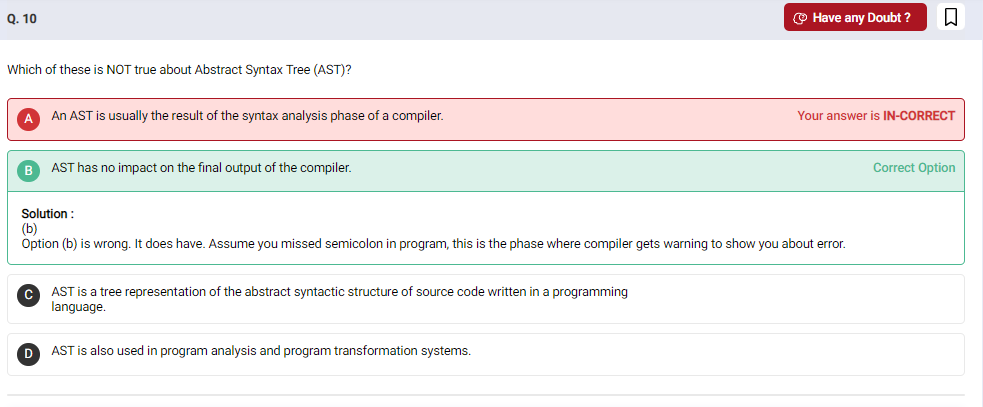




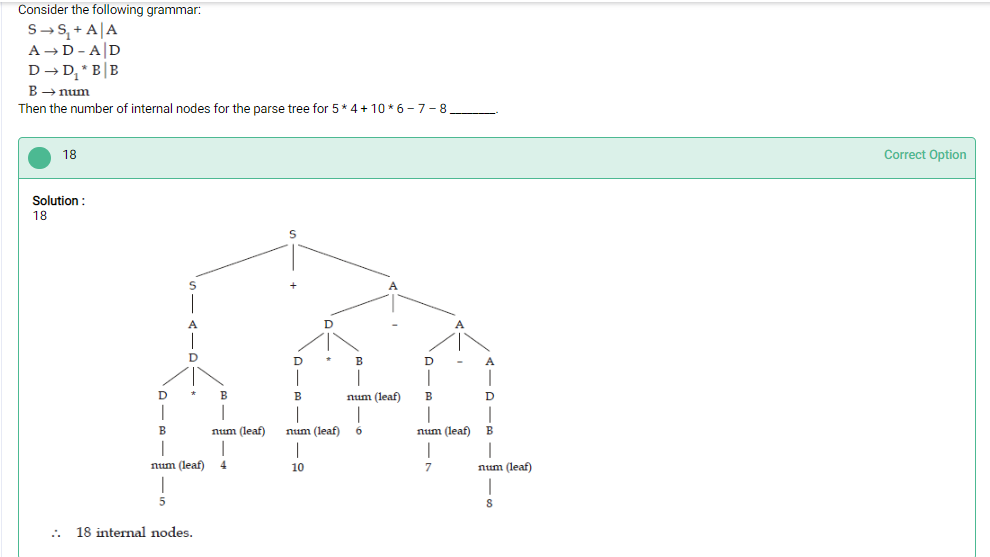
Abstract Syntax Tree is a kind of tree representation of the abstract syntactic structure of source code written in a programming language. Each node of the tree denotes a construct occurring in the source code.

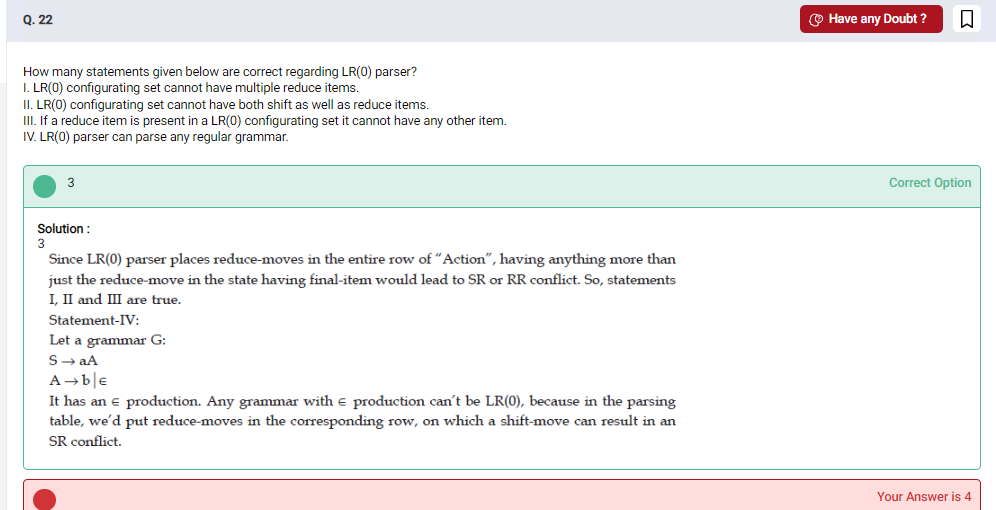
There is numerous importance of AST with application in compilers as abstract syntax trees are data structures widely used in compilers to represent the structure of program code. An AST is usually the result of the syntax analysis phase of a compiler. It often serves as an intermediate representation of the program through several stages that the compiler requires, and has a strong impact on the final output of the compiler.

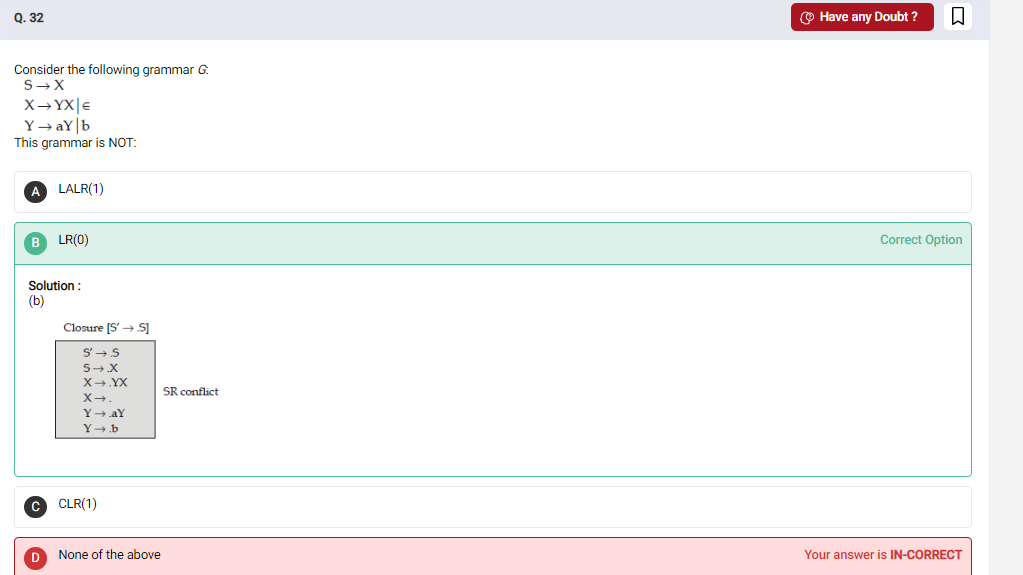
Let us do discuss the use of AST before proceeding further to the implementation part.AST’s are mainly used in compilers to check code for their accuracy. If the generated tree has errors, the compiler prints an error message. Abstract Syntax Tree (AST) is used because some constructs cannot be represented in context-free grammar, such as implicit typing. They are highly specific to programming languages, but research is underway on universal syntax trees.

 ABSTRACT SYNTAX TREE 

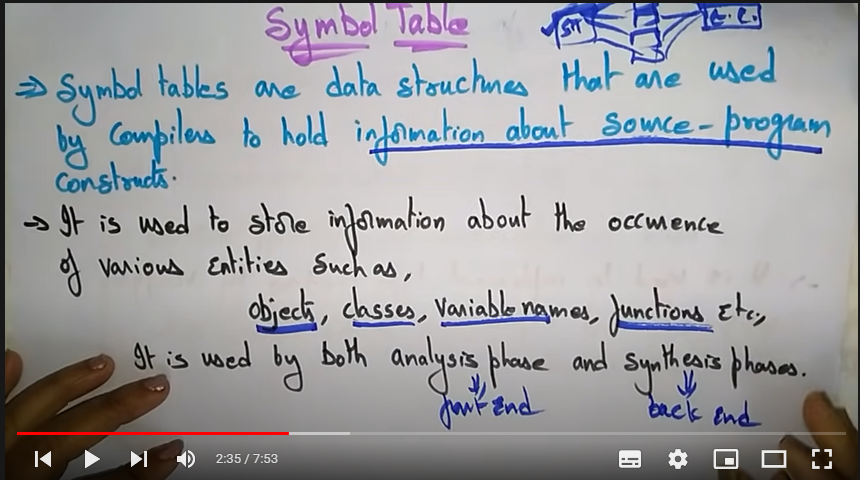
Intermediate code generator receives input from its predecessor phase, semantic analyzer, in the form of an annotated syntax tree. That syntax tree then can be converted into a linear representation, e.g., postfix notation. Intermediate code tends to be machine independent code. Therefore, code generator assumes to have unlimited number of memory storage (register) to generate code.

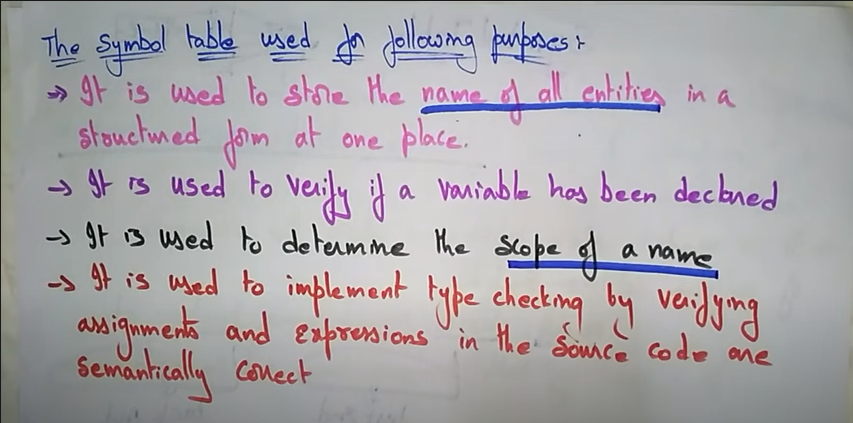


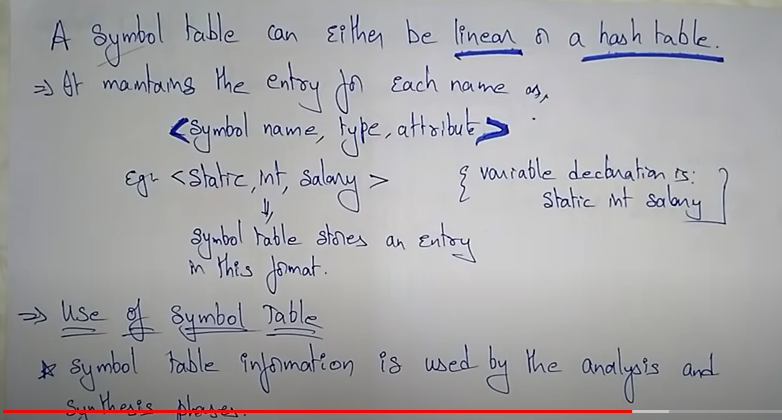


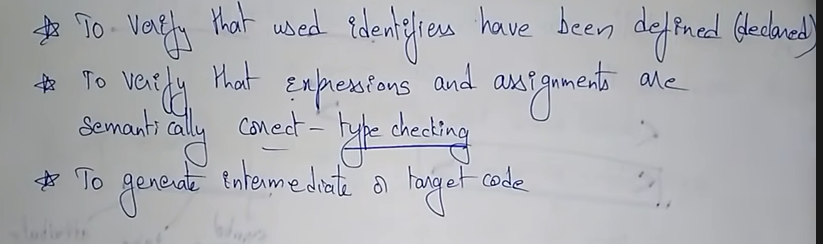


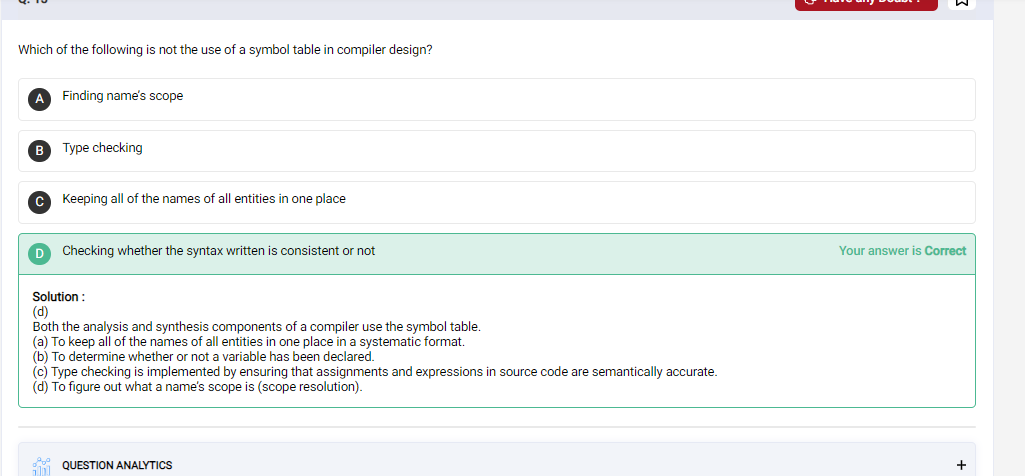


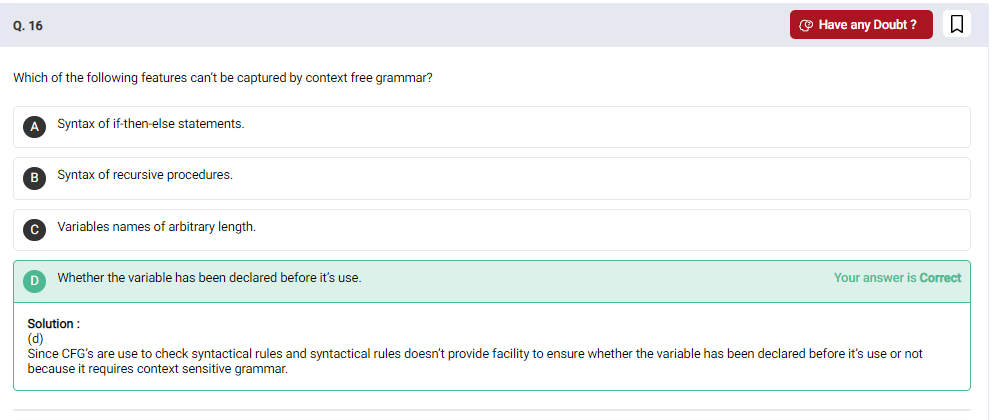


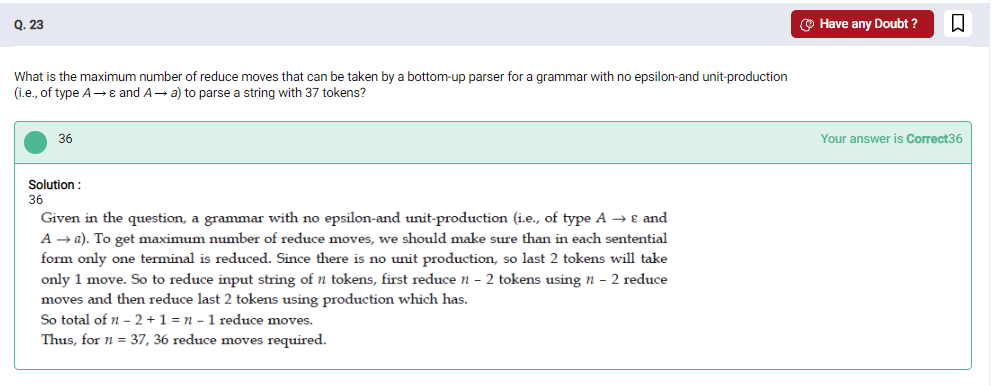












Go classes:

