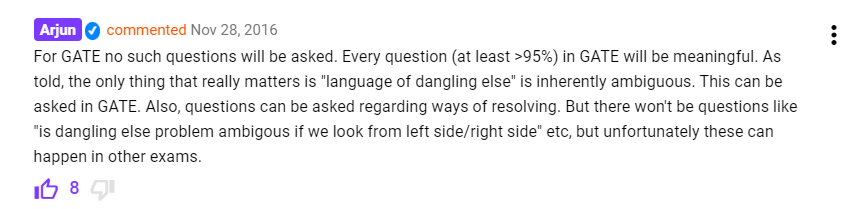
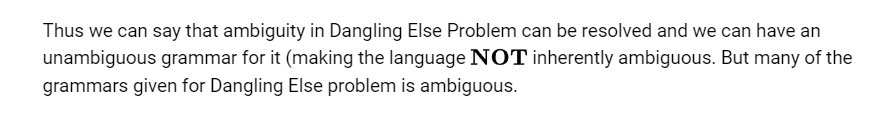
[compiler construction - Lexical and Semantic Errors in C - Stack Overflow](https://stackoverflow.com/questions/15570553/lexical-and-semantic-errors-in-c)

any one of heap and stack is enough to support recursion.  Dynamic allocation of activation records is essential to implement recursion. Remember the stack size can also grow dynamical (see C memory layout).

MUST MUST READ ABOUT AMBIGUITY, DCFL, Inherent ambiguity, LR(k): [Compiler Design: CAN Inherently ambiguous languages HAVE DETERMINISTIC CONTEXT FREE GRAMMARS? (gateoverflow.in)](https://gateoverflow.in/54718/inherently-ambiguous-languages-deterministic-context-grammars) and this comment too: <https://gateoverflow.in/54718/inherently-ambiguous-languages-deterministic-context-grammars?show=175871#c175871>

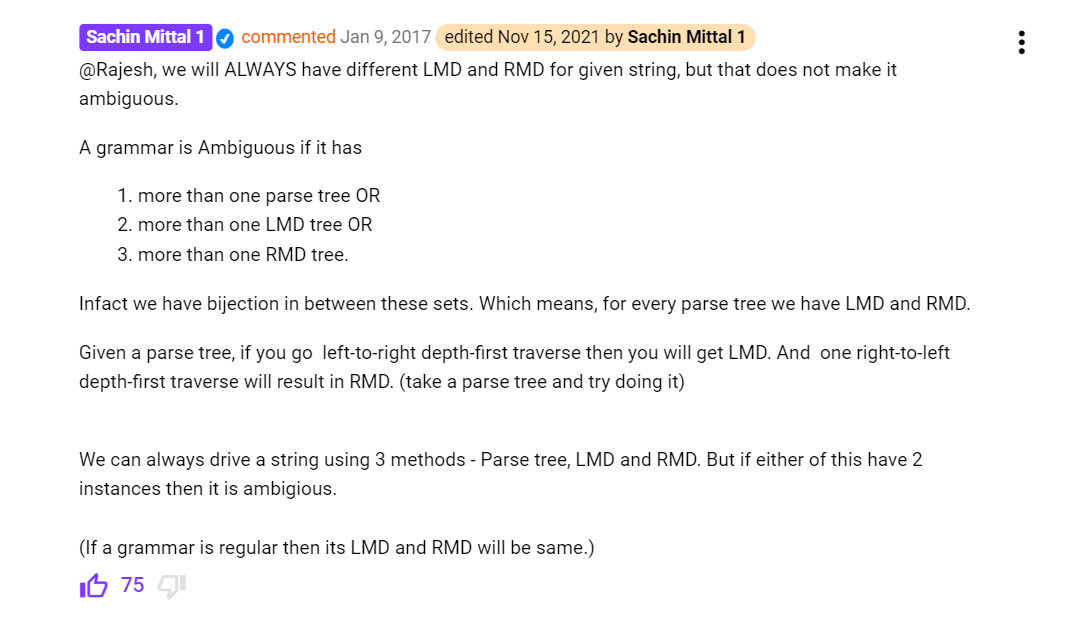




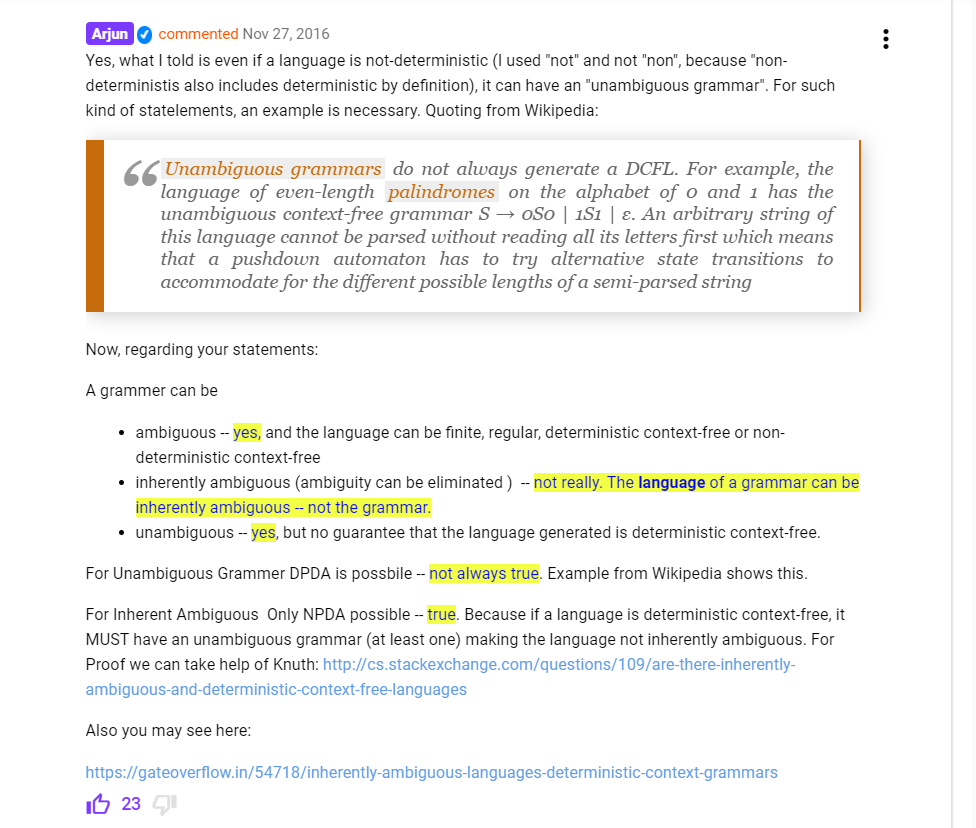
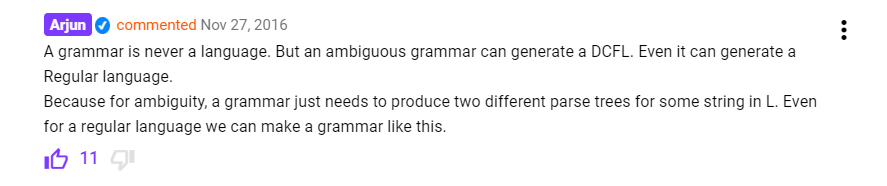
If a language is inherently ambiguous it can have no unambiguous grammar possible. Now, no DCFL is inherently ambiguous - any DCFL must have some unambiguous grammar.

[Compiler Design: GO Classes Test Series 2023 | Mock GATE | Test 2 | Question: 54 (gateoverflow.in)](https://gateoverflow.in/388654/go-classes-test-series-2023-mock-gate-test-2-question-54?show=392594#c392594)

Quickly revise my weak topic: [Chomsky Hierarchy in Theory of Computation - GeeksforGeeks](https://www.geeksforgeeks.org/chomsky-hierarchy-in-theory-of-computation/) [all type grammar rules]



predictive parse table = LL(1) table

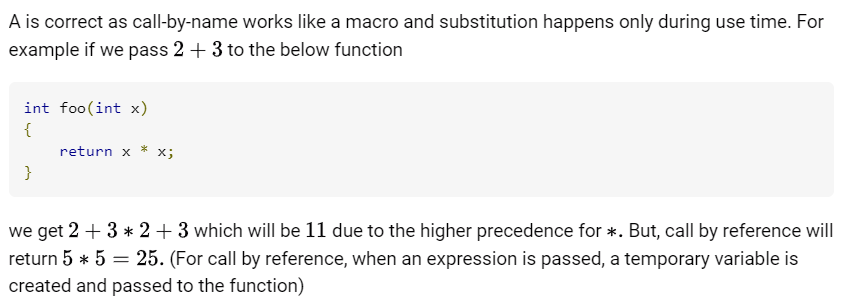


Value-result parameter passing was used in Fortran IV and in Ada. The idea is that, as for pass-by-value, the value (not the address) of the actual parameters are copied into the called method's AR. However, when the called method ends, the final values of the parameters are copied back into the arguments. Value-result is equivalent to call-by-reference **except** when there is aliasing (note: "equivalent" means the program will produce the same results, not that the same code will be generated).

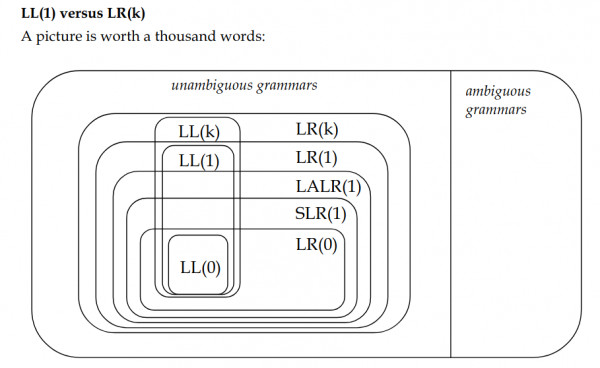
[Compiler Design: GATE CSE 1988 | Question: 8i (gateoverflow.in)](https://gateoverflow.in/94371/gate-cse-1988-question-8i) [About call by value result/copy restore]

For call by reference, when an expression is passed, a temporary variable is created and passed to the function

Whereas, in call by name, when an expression is passed, the expression is copied and used every where.



<https://gateoverflow.in/94371/gate-cse-1988-question-8i?show=392681#c392681>

\*\*\*\*\*\*\* V. Imp

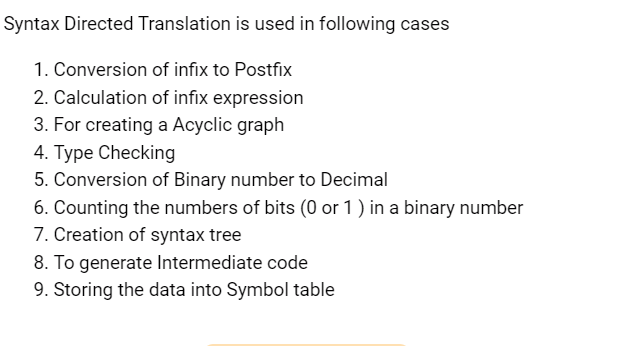
<https://gateoverflow.in/2339/gate-cse-2010-question-38?show=392686#c392686>

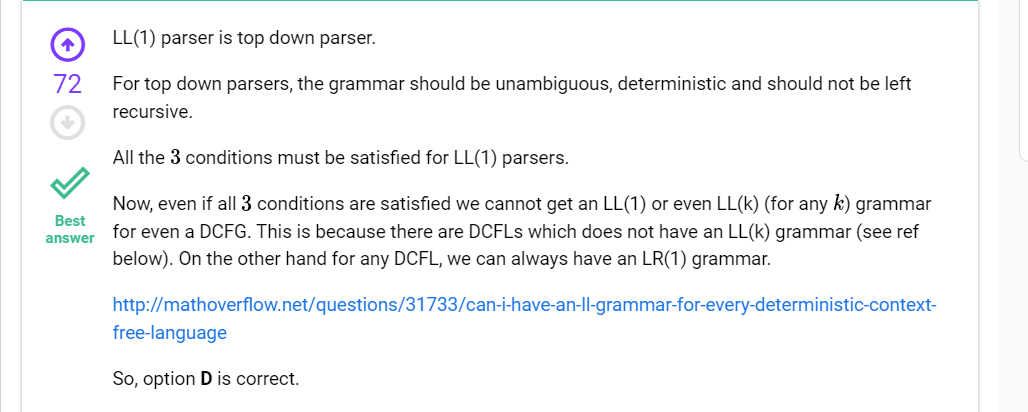
[Compiler Design: GATE CSE 1991 | Question: 09a (gateoverflow.in)](https://gateoverflow.in/536/gate-cse-1991-question-09a)

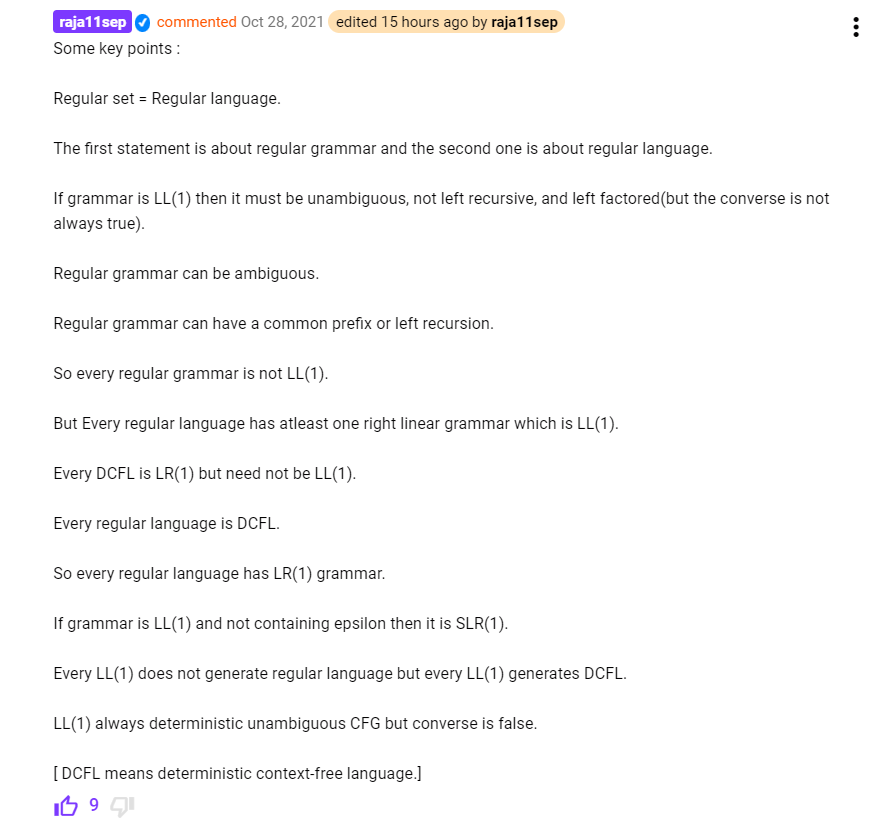
[programming languages - What is call-by-need? - Stack Overflow](https://stackoverflow.com/questions/5526059/what-is-call-by-need) [imp]

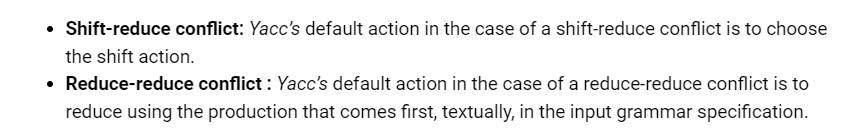
[Compiler Design: GATE CSE 1991 | Question: 09b (gateoverflow.in)](https://gateoverflow.in/43603/gate-cse-1991-question-09b)

[Compiler Design: GATE CSE 2003 | Question: 74 (gateoverflow.in)](https://gateoverflow.in/43575/gate-cse-2003-question-74) [God level ans by Deepak Sir]





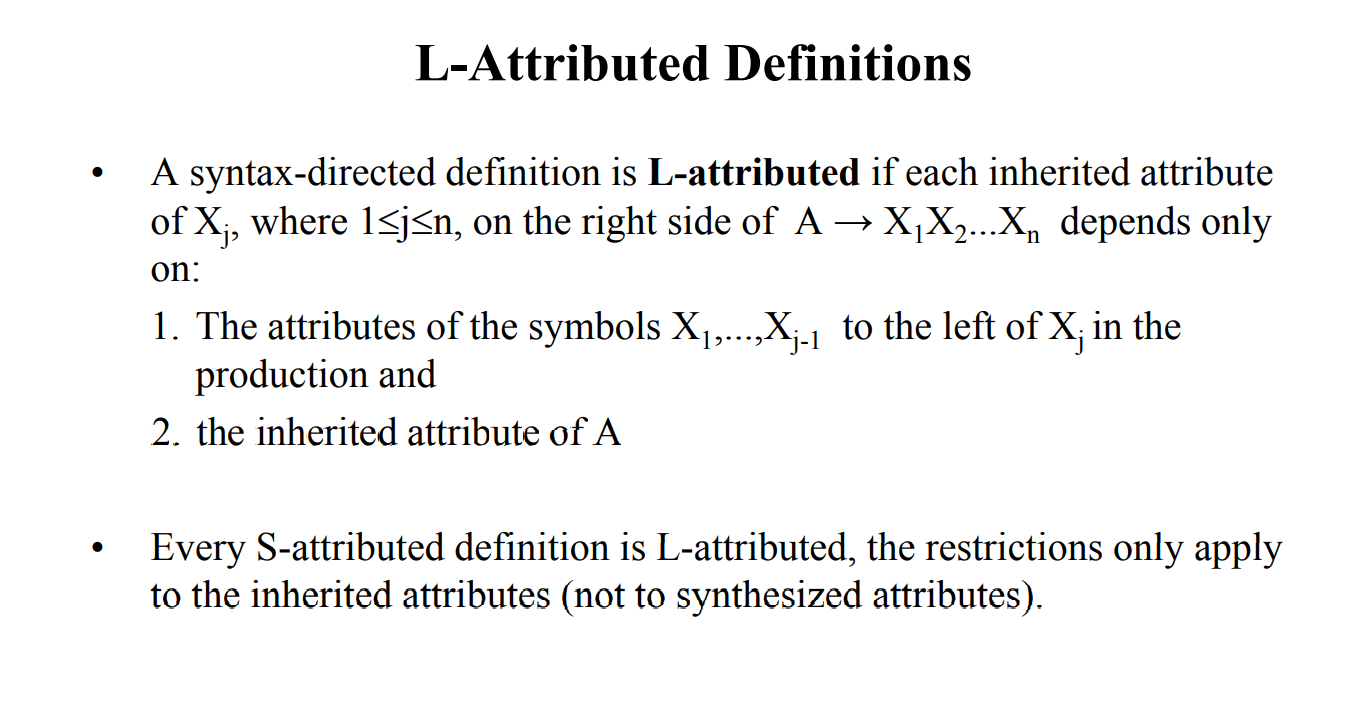


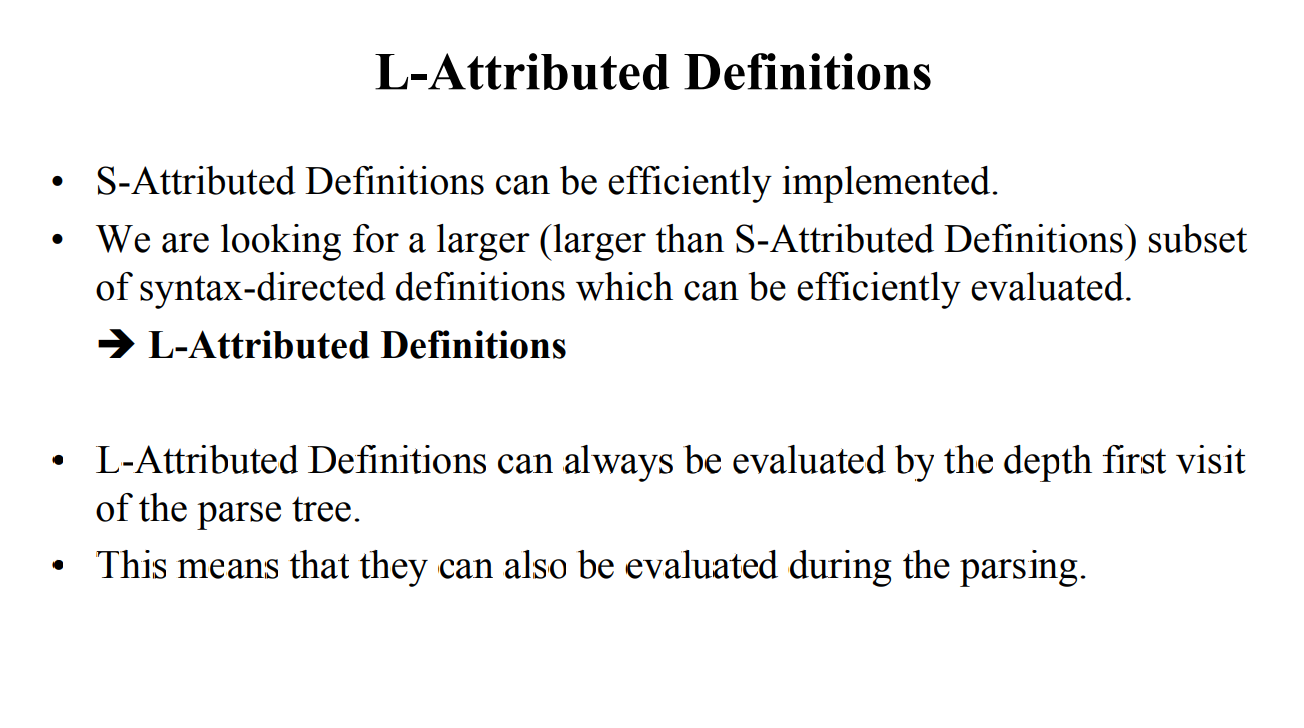


[Compiler Design: GATE CSE 2005 | Question: 83b (gateoverflow.in)](https://gateoverflow.in/87037/gate-cse-2005-question-83b)

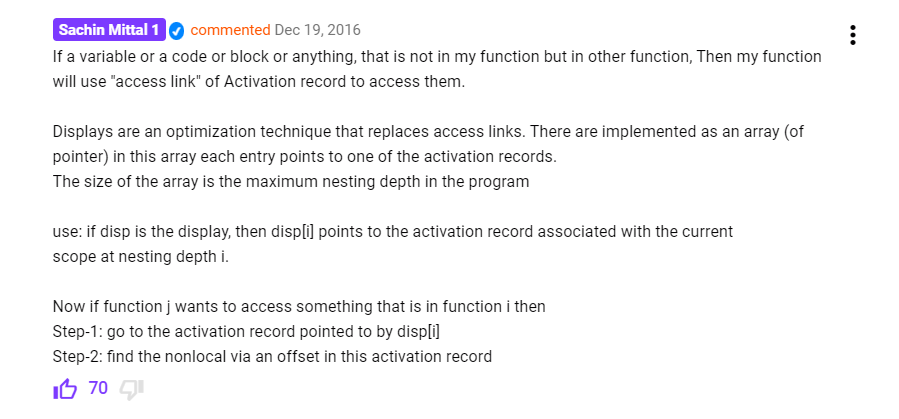
Handle vs Viable prefix: [Compiler Design: Handle in a grammar (gateoverflow.in)](https://gateoverflow.in/244291/handle-in-a-grammar?show=365479#c365479) , <https://gateoverflow.in/409/gate-cse-2008-question-11?show=334394#c334394>, <https://gateoverflow.in/409/gate-cse-2008-question-11?show=103537#c103537>

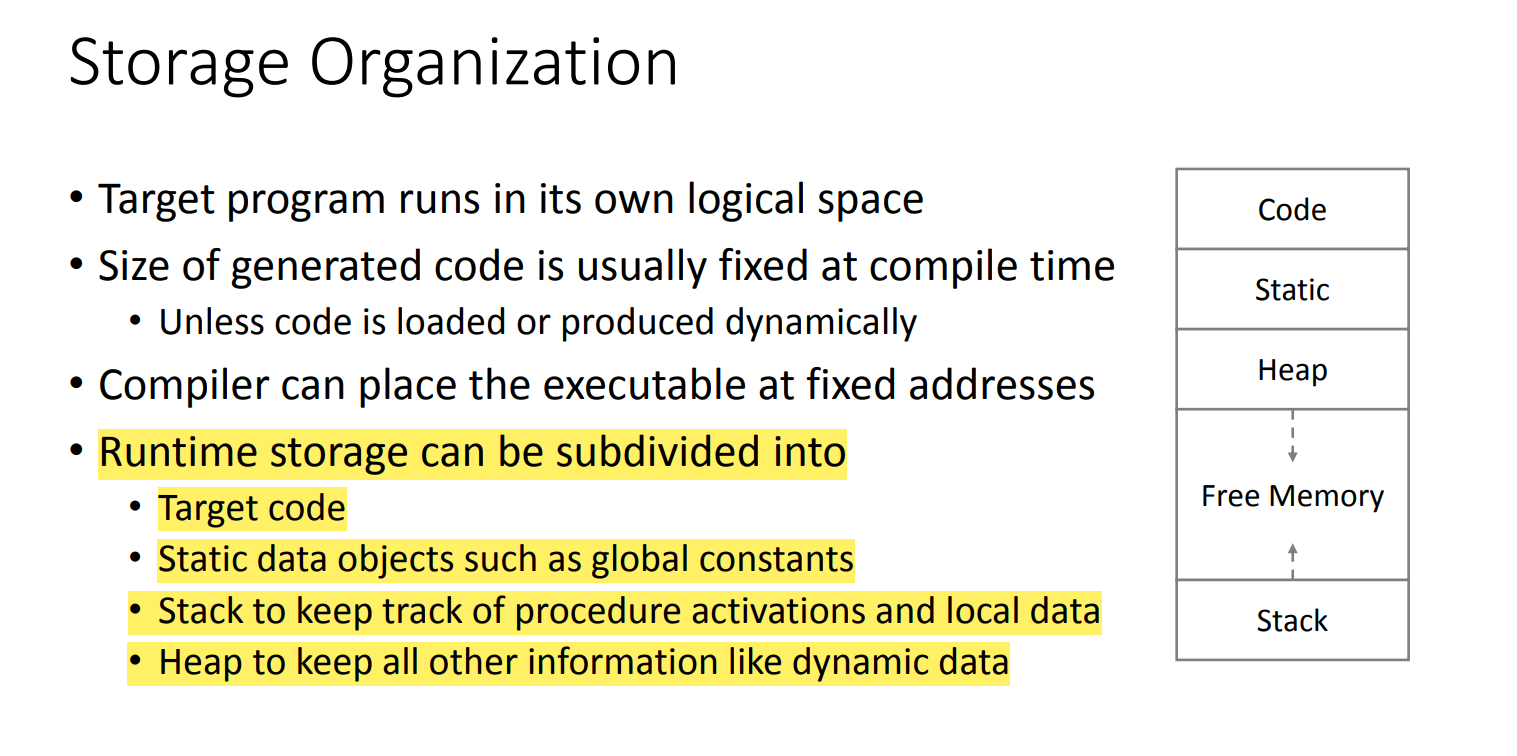
Number of reductions = (number of non-leaf nodes in the parse tree)

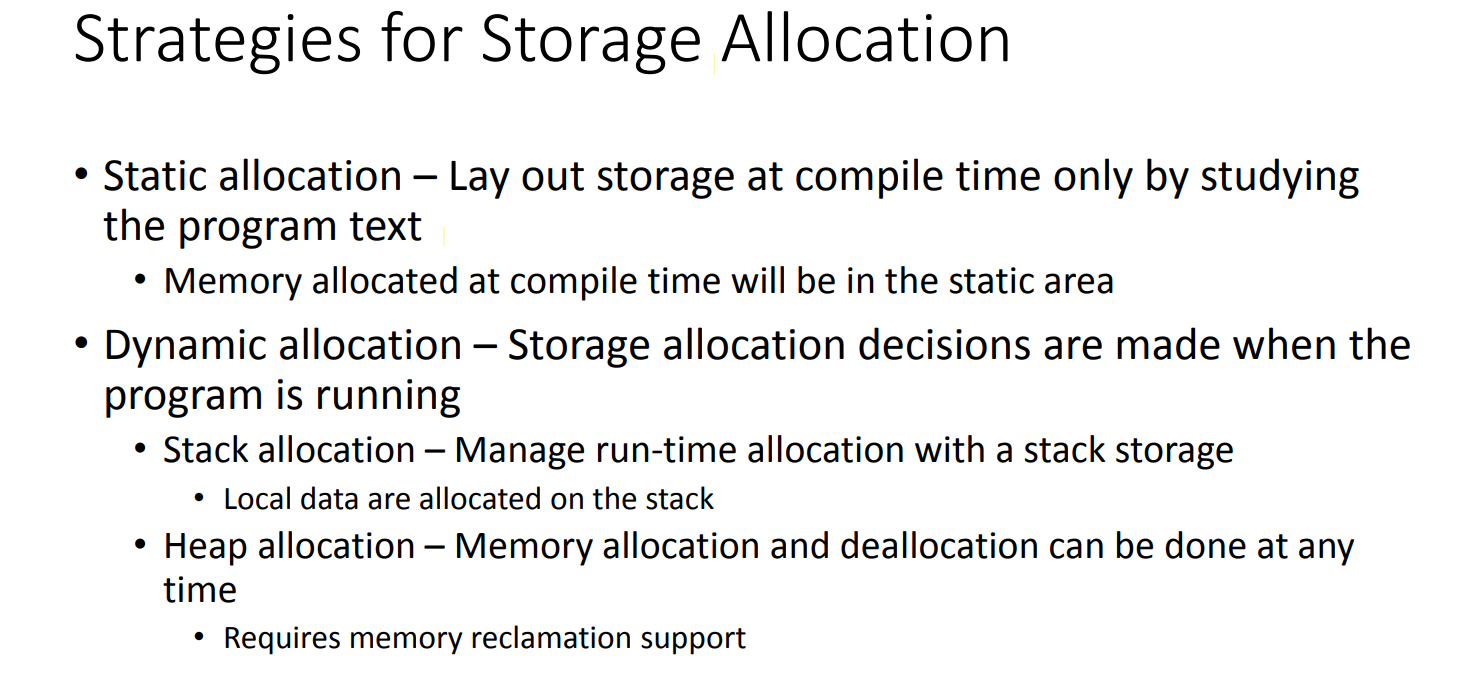


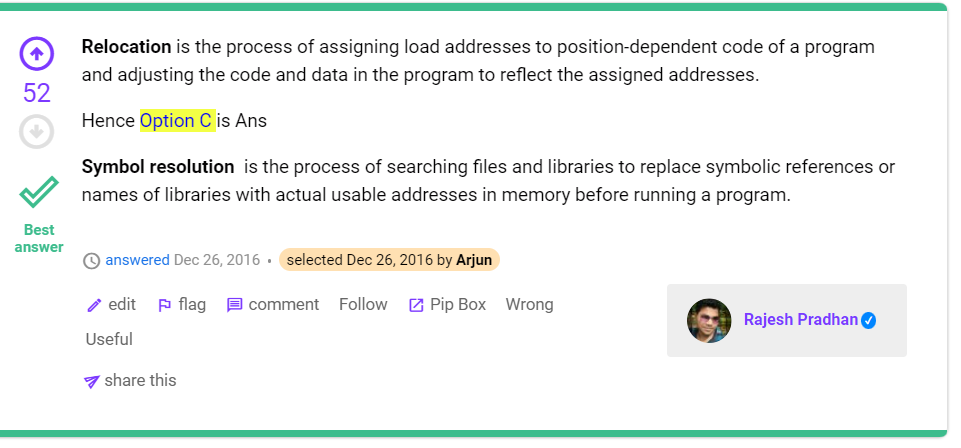


[PowerPoint Presentation (iitk.ac.in)](https://www.cse.iitk.ac.in/users/swarnendu/courses/spring2020-cs335/Runtime%20Environments.pdf) : Pg 63-66 (about displays) -> Must read all pages for runtime env

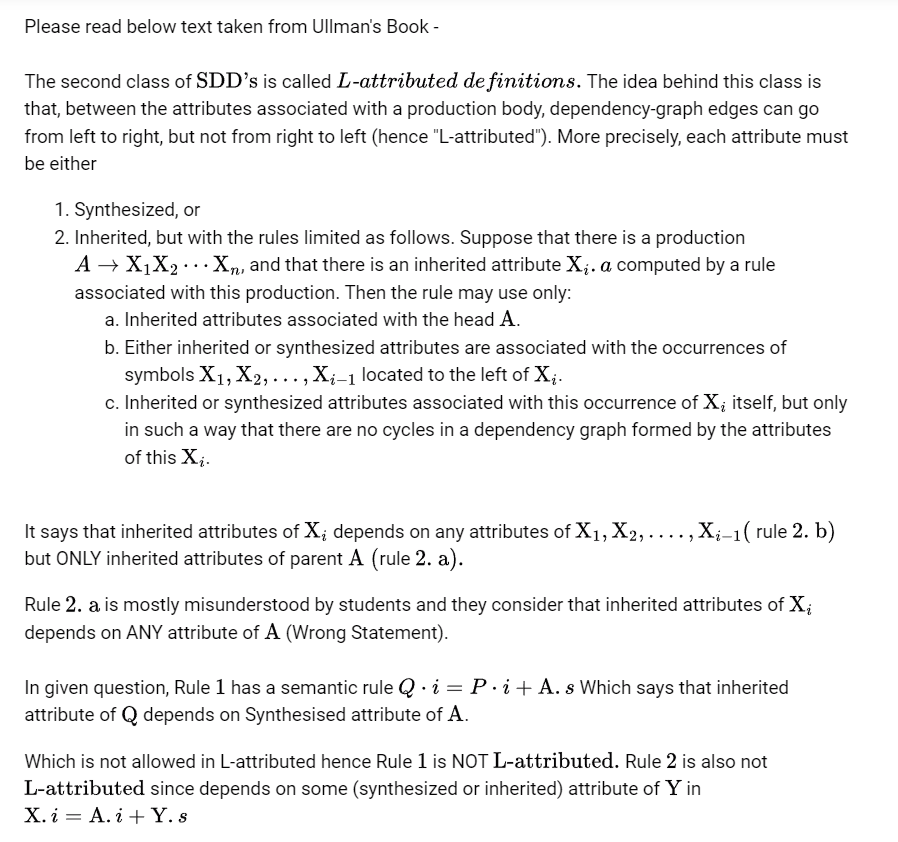


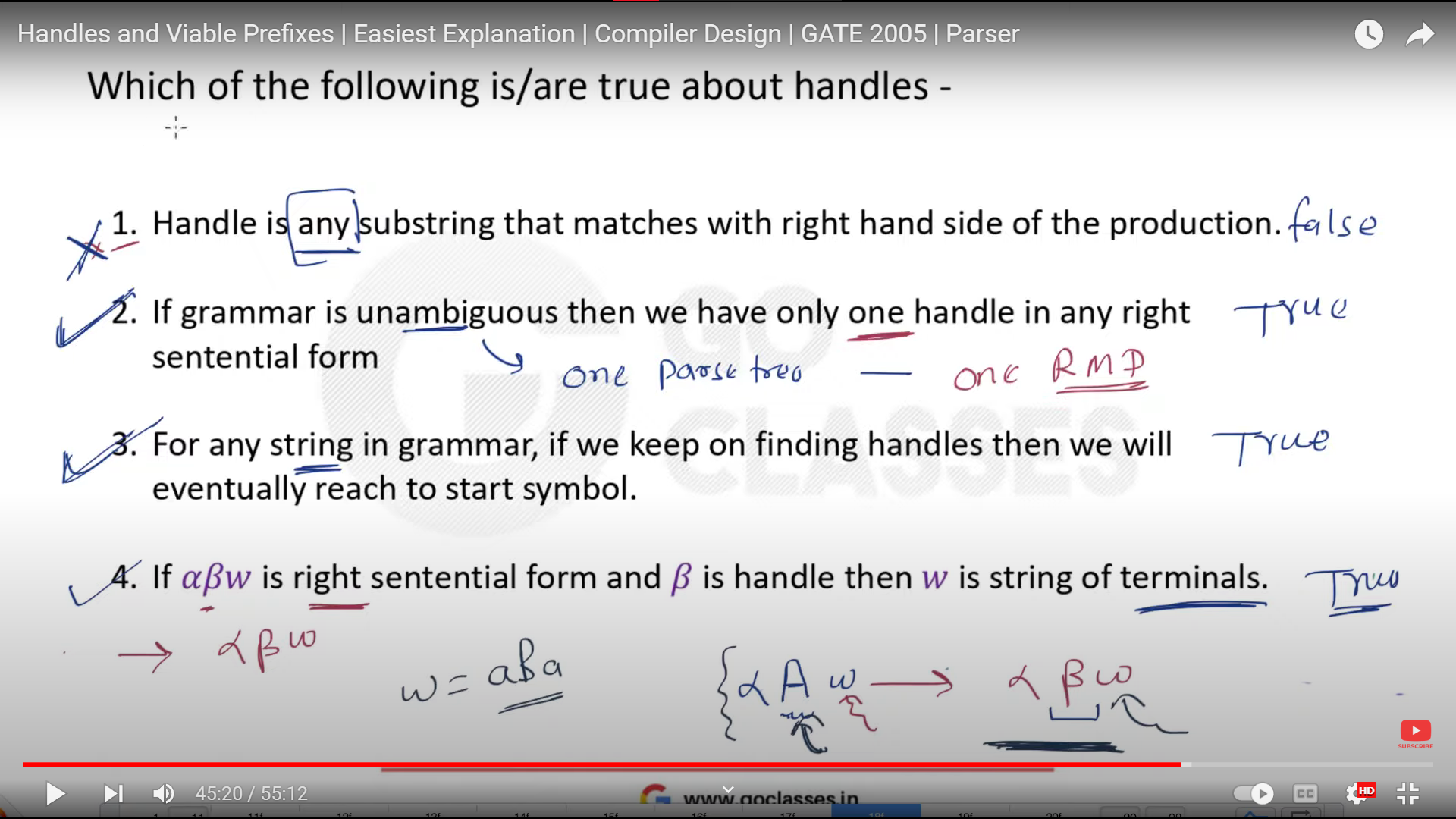


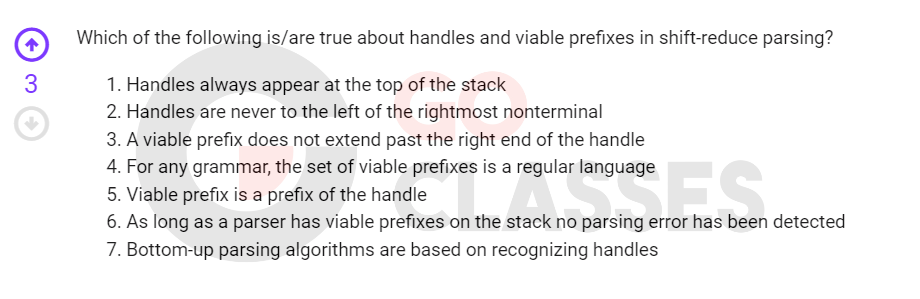




s







ANSWER: ALL 1-7 are TRUE