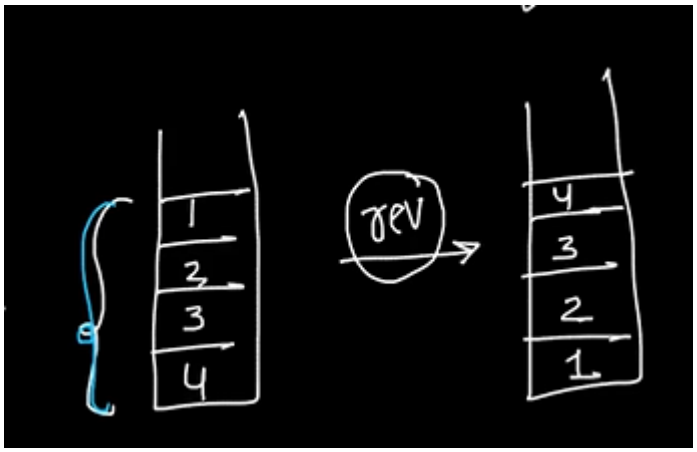
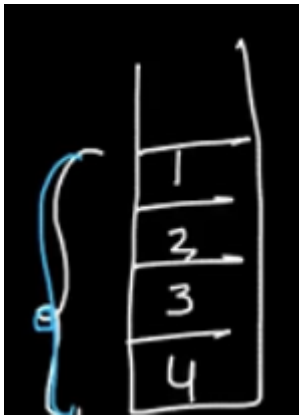
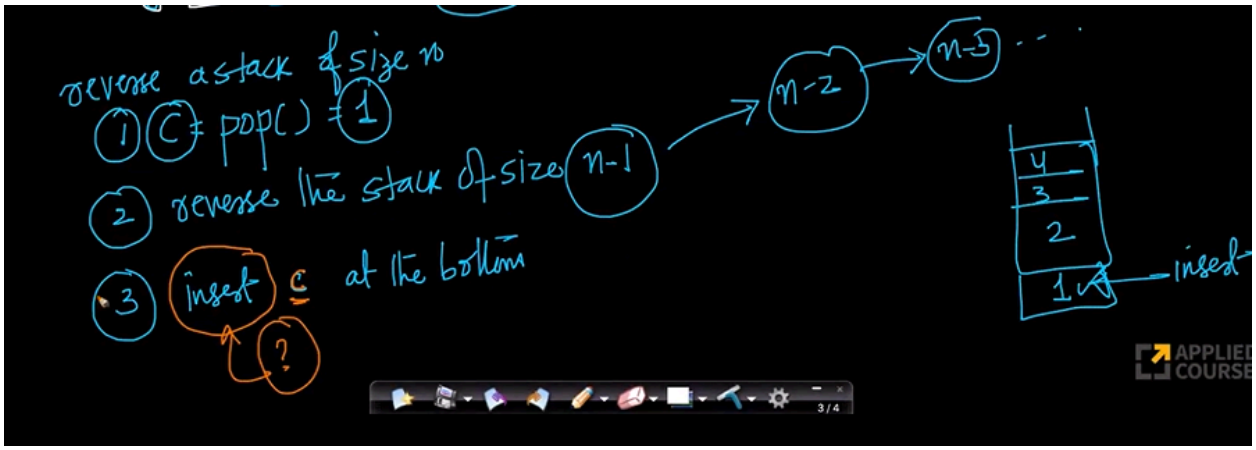
Reverse a stack

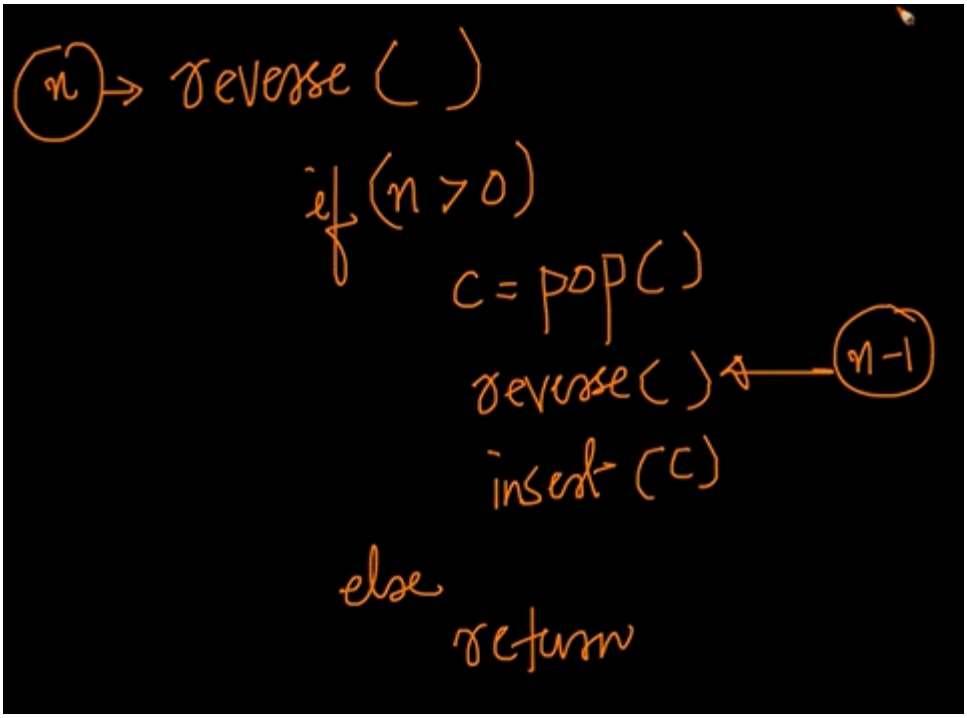
1. Our goal is 

2. What is recursion? To break down the problem into smaller sized and solve them
3. For any recursion problem..we have to handle 2 cases…one is to divide the problem into smaller pieces and other one is to handle boundary cases
4. Basic pseudocode
5. Lets suppose n = 4 

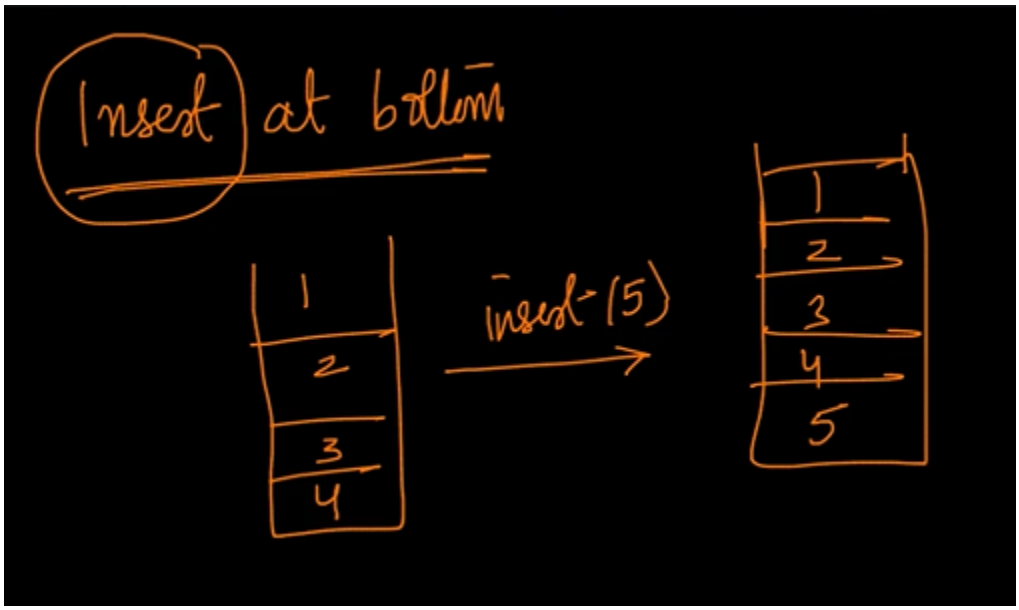
Now we'll perform reverse on n …in the reverse first we pop() top element and perform reverse on (n-1)



6. But the problem is we have to insert c at the bottom..how can we do that?
7. Here what would be the boundary case? Boundary case is n = 0 ..means stack is empty

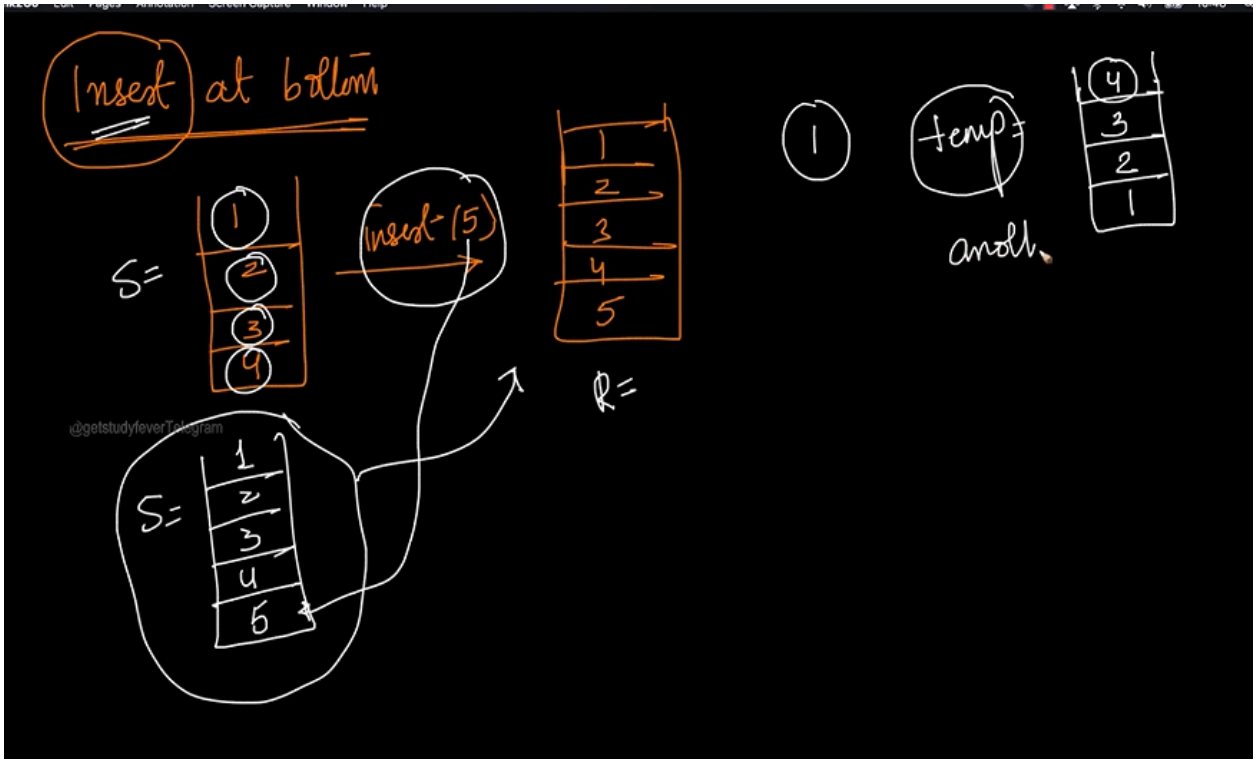
8. Pseudocode 

here if n>0 then ..we perform pop() ..then our n becomes n-1 and we call reverse() again…and later we insert(c) at bottom
9. Insert at bottom



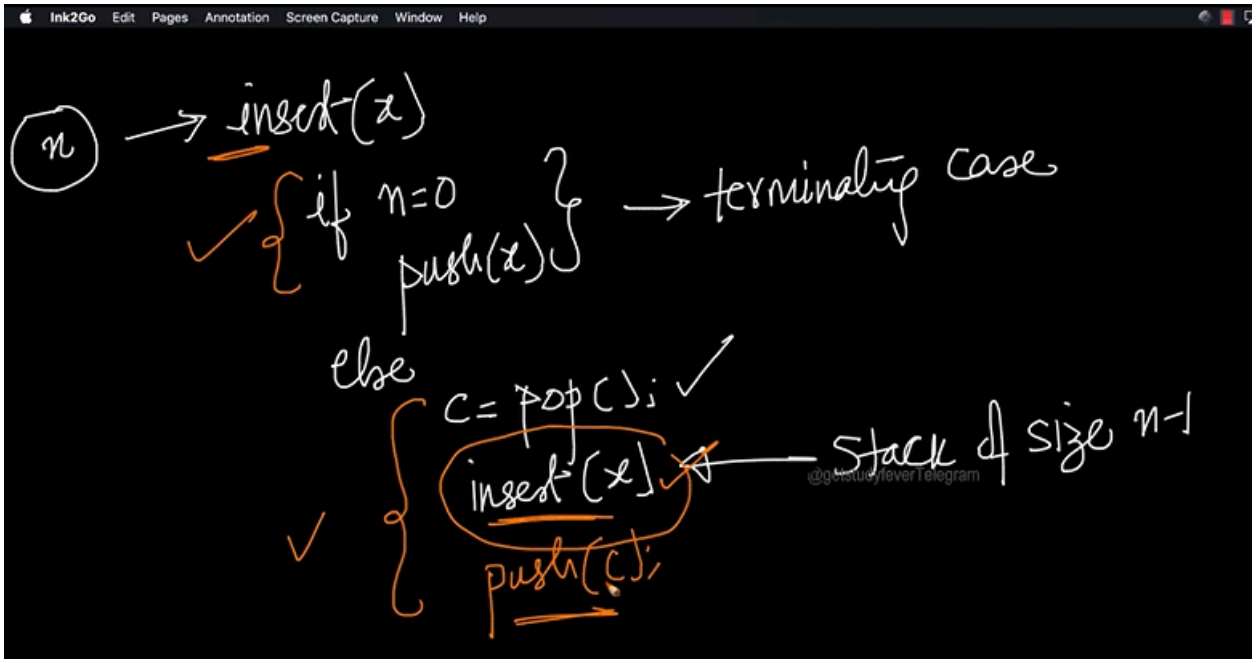
Here to insert 5…we have to pull 1 to 4 out and insert 5

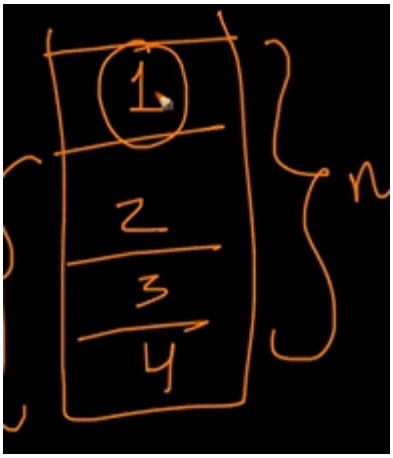
10. One way to insert at bottom of the stack is to use another stack



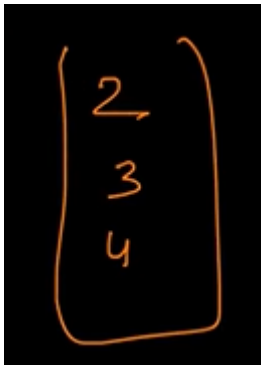
First we'll pop out from original stack and insert them in temp stack…now as the original stack is empty..we'll insert 5 at bottom…now again we'll pop from temp stack and insert in the original stack

11. Now lets look at recursive approach

12. 

Consider we are insert(5) 

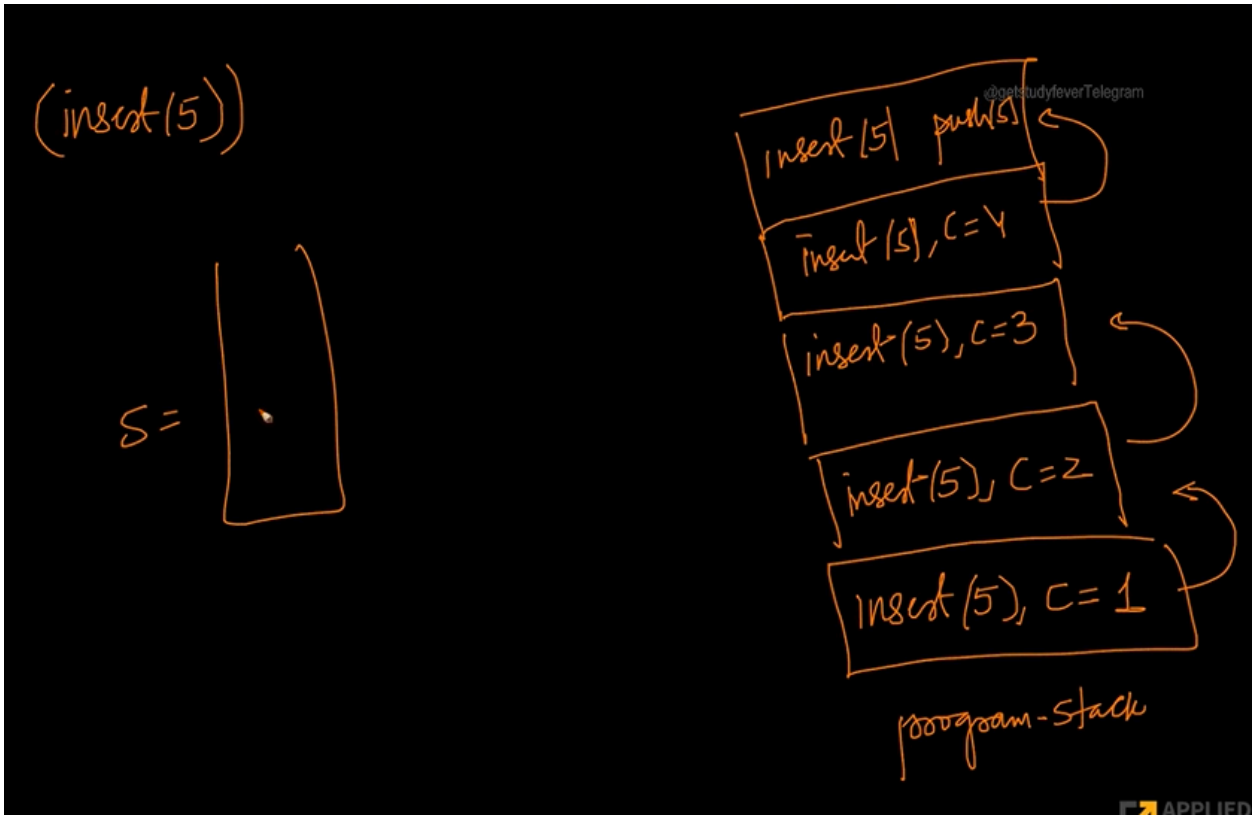
If n = 0…then we push(x) else we'll pop() from our original stack and store it in c

Again we will call insert(x) function now it contains 

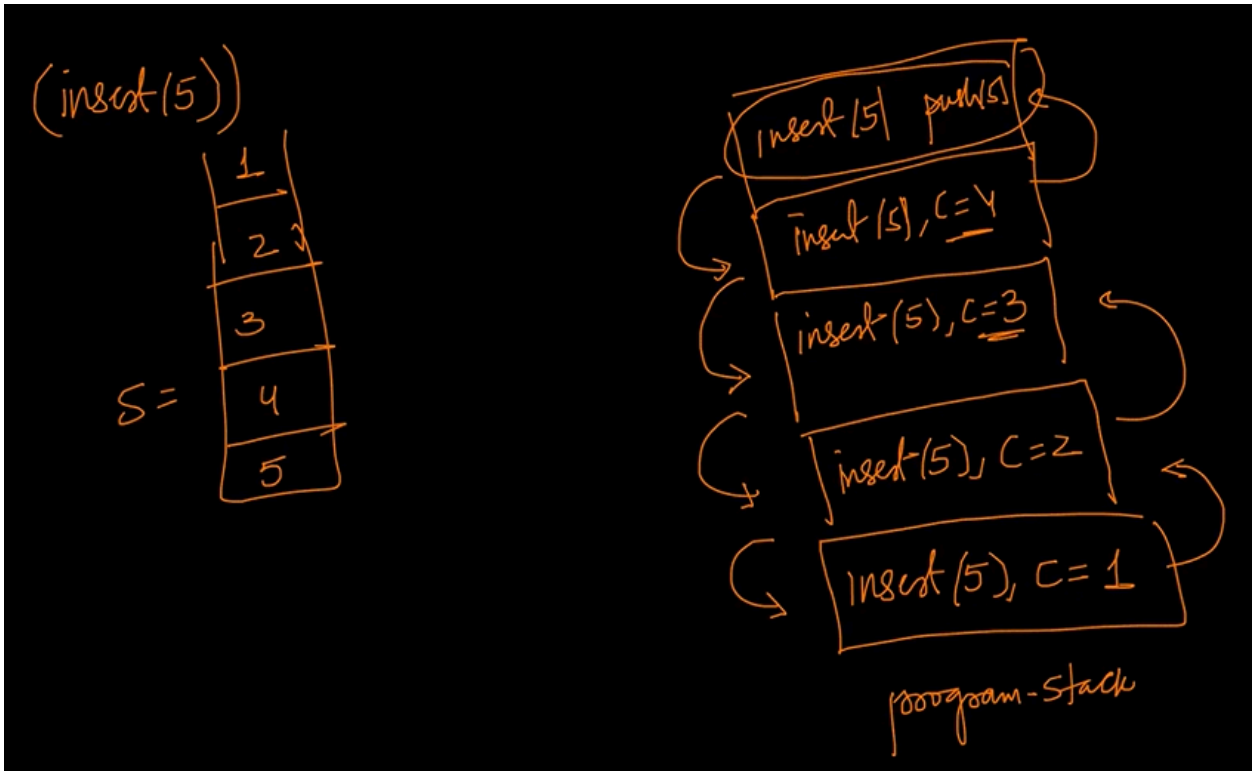
..in the end we push(c)

13. Lets c what happens internally

14. Recursion internally uses stack



when stack is empty we push(x)...later we push pop() element c



15. Here we'll be using two recursive function one for reversing and other for insert