9. Let *S* be an initially empty stack. We wish to perform two operations on S: *Add*(x) and *DeleteUntil* (x). These are defined as follows:

(a) *Add* (x): Add the element x to the top of the stack S. This operation takes O(1) time per invocation.

(b) *DeleteUntil* (x): Delete elements from the top of the stack up to and including the first x encountered. If p elements are deleted, the time taken is O(*p*).

Consider any sequence of *n* stack operations (*Adds* and *DeleteUntils*). Show how to amortize the cost of the *Add* and *DeleteUntil* operations so that the amortized cost of each is O(1). From this, conclude that the time needed to perform any such sequence of operations is O(n).

The worst case is that there are n-1 Add operation and the last is DeleteUntil, the amortize cost of per operation is $O\left(\frac{(n-1)\times 2}{n}\right) = O(1)$

=> which means any such sequence of operation need O(n) time.