Chapter-5:

R-5.3 Modify the experiment from Code Fragment 5.1 in order to demonstrate that Python’s list class occasionally shrinks the size of its underlying array when elements are popped from a list. **(Attach a screenshot of the output showing that, likewise, screenshot needs to include the day and time when it was taken)**

A screenshot of a computer

Description automatically generated

R-5.7 Let A be an array of size n ≥ 2 containing integers from 1 to n−1, inclusive, with exactly one repeated. Describe a fast algorithm for finding the integer in A that is repeated.

**There are a few different ways I’ve though of to do this. But the simplest would be to use a nested for loop in a for loop that will iterate through the array and compare every value of the array to the initial given value is being checked. Once it is found simply break from the loop. To improve the time complexity, ever time the initial for loop is iterated. The nested for loop only needs to check the values at the indexes plus 1 from the initial index. This way values that have already been checked to be equal will not be repeated.**

Chapter-6:

R-6.1 What values are returned during the following series of stack operations, if executed upon an initially empty stack? push(5), push(3), pop(), push(2), push(8), pop(), pop(), push(9), push(1), pop(), push(7), push(6), pop(), pop(), push(4), pop(), pop(). (remember what returns the value and what does not)

**In order: 3, 8, 2, 1, 6, 7, 4, 9**

* **I assume Pop returns the value in this example**

R-6.5 Implement a function that reverses a list of elements by pushing them onto a stack in one order, and writing them back to the list in reversed order. (have to be stack)

def reverse\_list(list):  
 temp = Stack()  
   
 for i in list:  
 temp.push(i)  
   
 list.clear()  
   
 while not temp.is\_empty():  
 list.append(temp.top())  
 temp.pop()

C-6.23 Suppose you have three nonempty stacks R, S, and T. Describe a sequence of operations that results in S storing all elements originally in T below all of S’s original elements, with both sets of those elements in their original order. The final configuration for R should be the same as its original configuration. For example, if R = [1,2,3], S = [4,5], and T = [6,7,8,9], the final configuration should have R = [1,2,3] and S = [6,7,8,9,4,5].

**To keep the same order you’d need to move each element exactly twice. Doing it once will result in the order being reversed.**

**So firstly, pop each element of T into R (R = [1,2,3,9,8,7,6] and T = []), and then pop each element of S into T (T = [5,4] and S = []). Finally pop each element of R that existed in T into S (This moved every element of T twice which means it is now in its original order. S = [6,7,8,9] and R = [1,2,3]). Now just move the values in T which were originally in S back into S ( S = [6,7,8,9,4,5], T = [] ). R isn’t changed in the final configuration.**