Describe the run-time bounds for last weeks tasks using bigO notation

Task 1 – My Code

from random import choice (1)

def listCreation(x): (1)

myList = [] (1)

i=0 (1)

while i < listQuantity: (N)

listItem = int(input("Enter list item ")) (N)

myList.append(listItem) (N)

i=i+1 (N)

print("Your selected list: " + str(myList)) (1)

Shuffler(myList) (1)

def Shuffler(theList): (1)

list\_length = len(theList) (1)

newList = [] (1)

for i in range(list\_length): (N)

element = choice(theList) (N)

newList.append(element) (N)

theList.remove(element) (N)

print(newList) (1)

listQuantity = int(input("Enter the number of elements in the list ")) (1)

listCreation(listQuantity) (1)

From my code above, we can see that some lines of code run once, while the loops run N number of times. Altogether the code has run time of 8N + 12. When we remove the constants and the multiplier we would write the run time as O(n), in bigO notation.

Task 2 – My Code

def factorial(myNumber): (1)

workingValue = 1 (1)

while myNumber >= 1: (N)

workingValue = workingValue \* myNumber (N)

myNumber = myNumber – 1 (N)

print("Your factorial is " + str(workingValue)) (1)

trailing(workingValue) (1)

def trailing(factorialResult): (1)

count = 0 (1)

factorialResult = str(factorialResult) (1)

for c in reversed(factorialResult): (N)

c = int(c) (N)

if c == 0: (N)

count = count + 1 (N)

else: (N)

break (N)

print ("Number of trailing 0's is " + str(count)) (1)

originalNumber = int(input("Enter a number ")) (1)

num = factorial(originalNumber) (1)

For the code above, again, we can see that some of the lines of code execute only once while the loops execute N times. Altogether the run time for this program is 9N + 10. Once the constant and the multiplier is removed, we find that the bigO representation of the run time is O(n).