**Question 1:**

ARRAYMAX(A, N)

* 1. CURRENTMAX <- A[0]
  2. FOR I<-1 TO N-1 DO
  3. IF A[I] > CURRENTMAX THEN
     1. CURRENTMAX <- A[I]
  4. {INCREMENT COUNTER I}
  5. RETURN CURRENTMAX

Looking at this code segment, we can see that the only loop is on line b. We see that the loop goes for the length of the array, and everything else is constant, so the time complexity would be **O(n)**.

**Question 2:**

PREFIXAVERAGES1(X,N)

* 1. INPUT: ARRAY X OF N INTEGERS
  2. OUTPUT: ARRAY A OF PREFIX AVERAGES OF X
  3. A <- NEW ARRAY OF N INTEGERS
  4. FOR I <- 0 TO N – 1 DO
     1. S <- X[0]
     2. FOR J <- 1 TO I DO
        1. S <- S + X[J]
        2. A[I] <- S/(I+1)

RETURN A

This code segment has two loops. The outer loop on line c loops the length of the array. The inner loop only goes to whatever the current place of i is. This means that in the *worst case*, which is what big O represents, it will also loop the full complement of the array. This means that this algorithm is **O(n2)**.

**Question 3:**

PREFIX AVERAGES2(X,N)

INPUT: ARRAY X OF N INTEGERS

OUTPUT: ARRAY A OF PREFIX AVERAGES OF X

A <- NEW ARRAY OF N INTEGERS

S<- 0

FOR I<- 0 TO N-1 DO

S<-S + X[I]

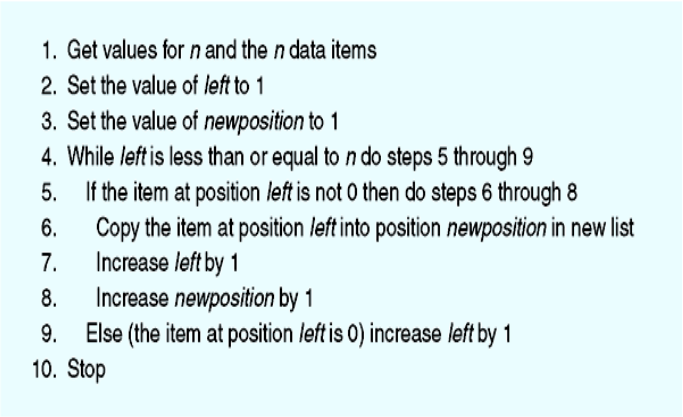
A[I] <- S/(I+1)

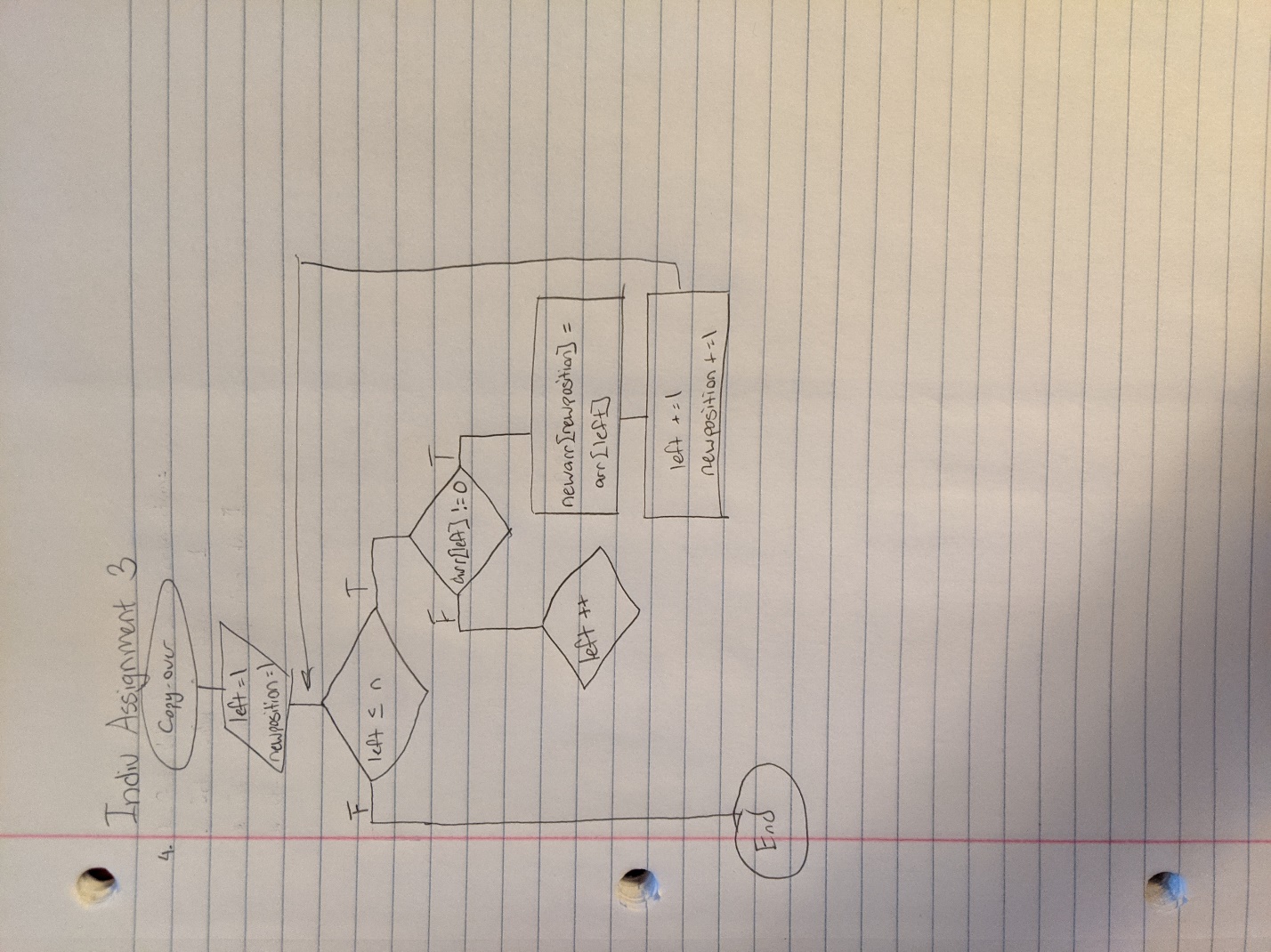
RETURN A

This segment only has one loop that spans the length of the array, so it’s **O(n)**.

**Question 4:**

Please draw a flowchart to explain the Copy-over algorithm for data cleanup.





**Question 5:**

Please draw a flowchart to explain the Converging-pointers algorithm for data cleanup.

