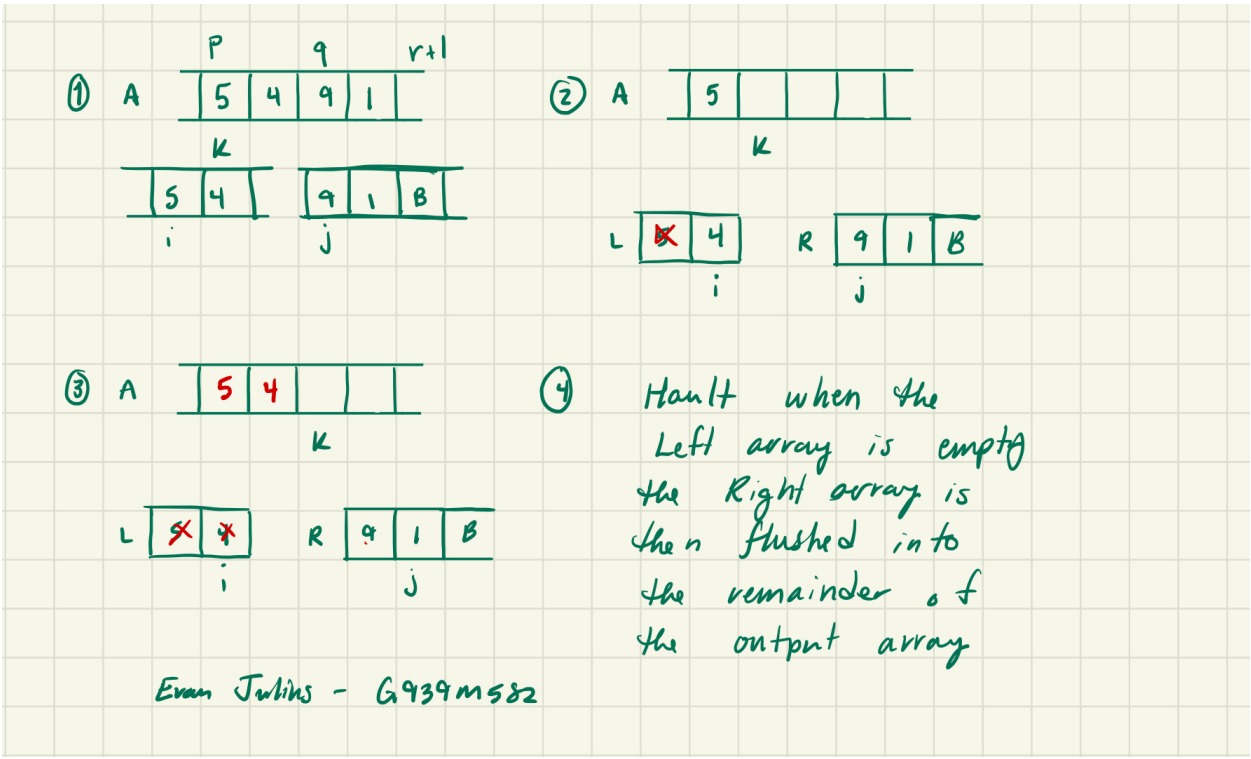
Evan Julius (G939M582)

Note: I was going to use Jupyter, but I wasn't sure how to go about including images.. Sorry!

Question 1

Please run the given buggy merge sort on array [5, 4, 9, 1] and demonstrate the merge process via a figure (like CLRS book Fig 2.3). Briefly state the bug and its fix (covered in 01/30's lecture.)



Question 2

For `arr = [x for x in range(1<< 10, 0, -1)]`:

1) Pass the array to an insertion sort function and record the total number of iterations the inner while loop executes.
2) Pass the array to a merge sort function (not the buggy one) and record the total number of iterations the for loop executes.
3) Change the array to `arr = [x for x in range(1<< 12, 0, -1)]` and repeat the above two steps. Explain the change in terms of O(n * n) and O(n * lg n).

1. 523776 times

2. 10240 times

3. For the insertion sort: 8386560; and for the merge sort: 49152 times. The change seen between the sorts increase exponentially or logarithmically. The insertion sort grows proportional to the square number of items in the list, while the merge sort grows proportional to the number of items in the list multiplied by the logarithm of the number of items in the list.

2-2 Correctness of Bubblesort

Bubblesort is a popular, but inefficient, sorting algorithm. It works by repeatedly swapping adjacent elements that are out of order.

```
BUBBLESORT(A)
1  for i = 1 to A.length - 1
2      for j = A.length downto i + 1
3          if A[j] < A[j - 1]
4              exchange A[j] with A[j - 1]
```

a. Let A' denote the output of BUBBLESORT(A). To prove that BUBBLESORT is correct, we need to prove that it terminates and that

$$A'[1] \le A'[2] \le \cdots \le A'[n] , \tag{2.3}$$

where $n = A.length$. In order to show that BUBBLESORT actually sorts, what else do we need to prove?

The next two parts will prove inequality (2.3).

b. State precisely a loop invariant for the **for** loop in lines 2–4, and prove that this loop invariant holds. Your proof should use the structure of the loop invariant proof presented in this chapter.

c. Using the termination condition of the loop invariant proved in part (b), state a loop invariant for the **for** loop in lines 1–4 that will allow you to prove inequality (2.3). Your proof should use the structure of the loop invariant proof presented in this chapter.

d. What is the worst-case running time of bubblesort? How does it compare to the running time of insertion sort?

a. The values of A' must consist of all the elements from A in ascending order.

b. At the start of the for loop, the subarray A' consists of the elements originally in A before the loop but in a different order while the first element is remains the smallest of all the elements in the list.
   - Initialization: The subarray contains only the smallest element from the original array
   - Maintenance: In each loop, A and A' will be compared and the smallest element among them will be moved into A'
   - Termination: The loop will terminate when the index j is equal to zero. The inner loop is allowed one more loop to allow for the final elemental comparison.

c. At the start of each loop of the outer loop, A' will only consist of numbers that are smaller than the current elements in A.
   - Initialization: The A' is empty and the inner loop will start at the 2nd index of the original array
   - Maintenance: At the beginning of the outer loop, the array A' will contain the all the elements smaller that A but in sorted order. The inner loop will execute and add the next smallest element.
   - Termination: The loop will terminate when for loop has reached the end at A.length, and the array A will consist of all elements but in sorted order.

d. Both sorts shared the same running-time, but the bubble sort may have a larger cost factors than the insertion sort.