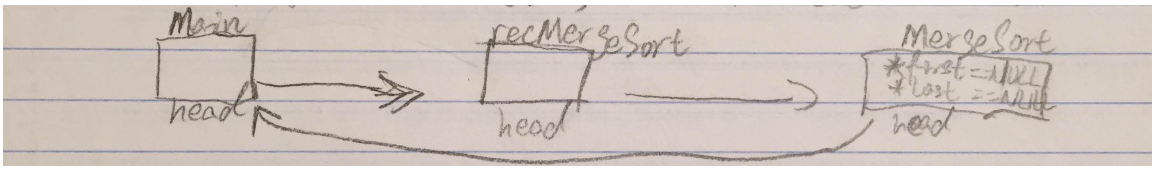
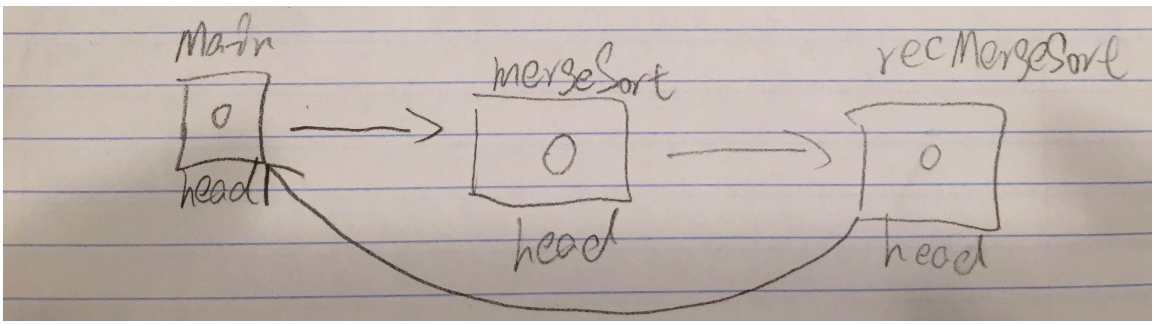
Xilun Guo
Final Project

Input Values	Expected Output	Did Actual Meet Expected?
Empty	Nothing	Yes
0	0	Yes
a, b, c	Nothing	Yes
23, 32, 42, 24	23 24 32 42	Yes
101, 99, 100	99 100 101	Yes
2, -2	-2 2	Yes
1 -1 0	-1 0 1	Yes
-3, -5, -10, -2	-10 -5 -3, -2	Yes
3, 3, -3	-3, 3, 3	Yes
10, 0,20	0 10 20	Yes

Test case 1: When the list is empty, *head in reMergeSort is NULL and the first check if head is NULL. Ie: if (head != NULL). Then go back to the main.

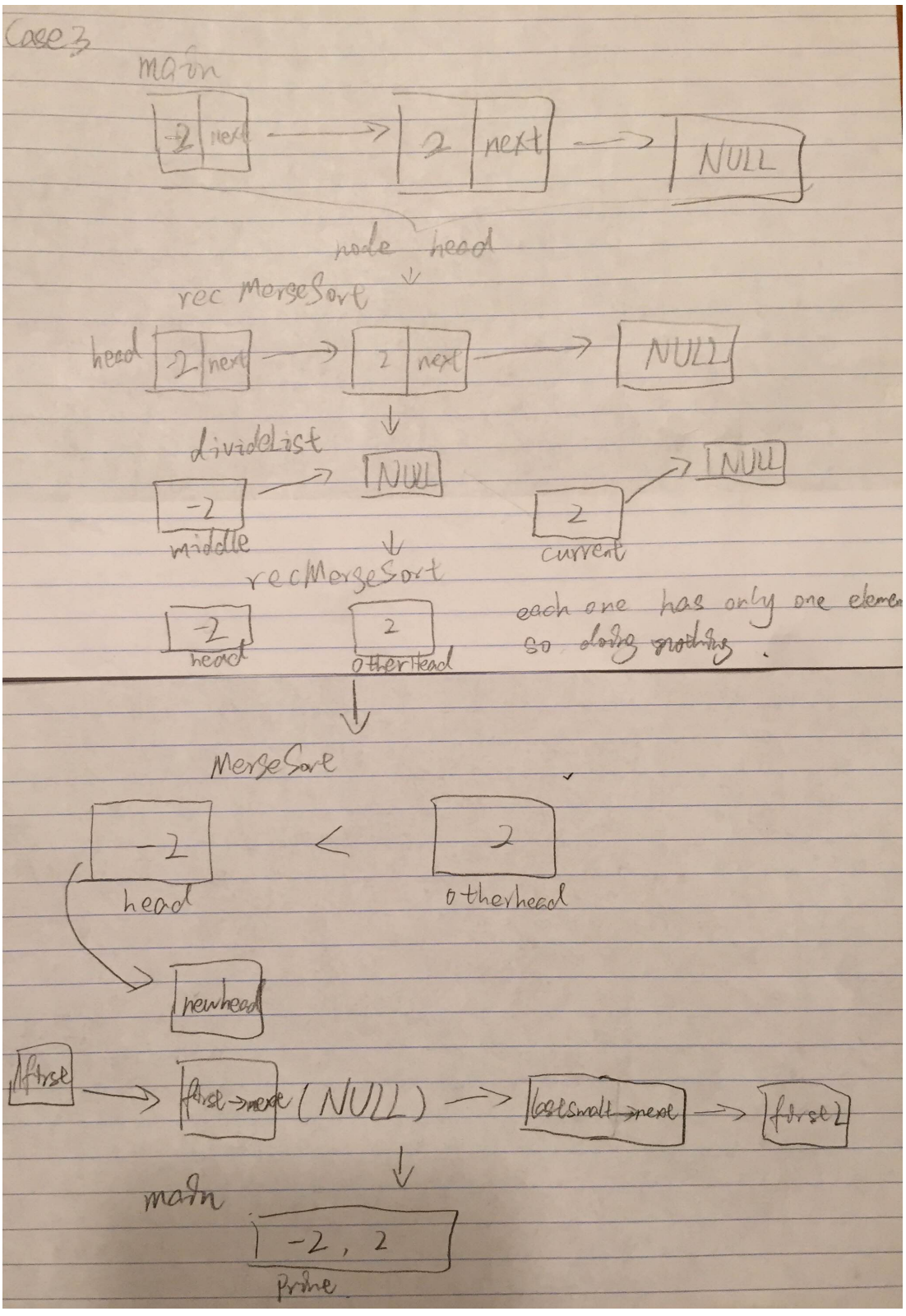


Test case 2: When the user just input 1 number to the list, for example “0”. It call recMergeSort function first, and check the list is not empty. However the next is empty, so back to main, and just print out “0”.



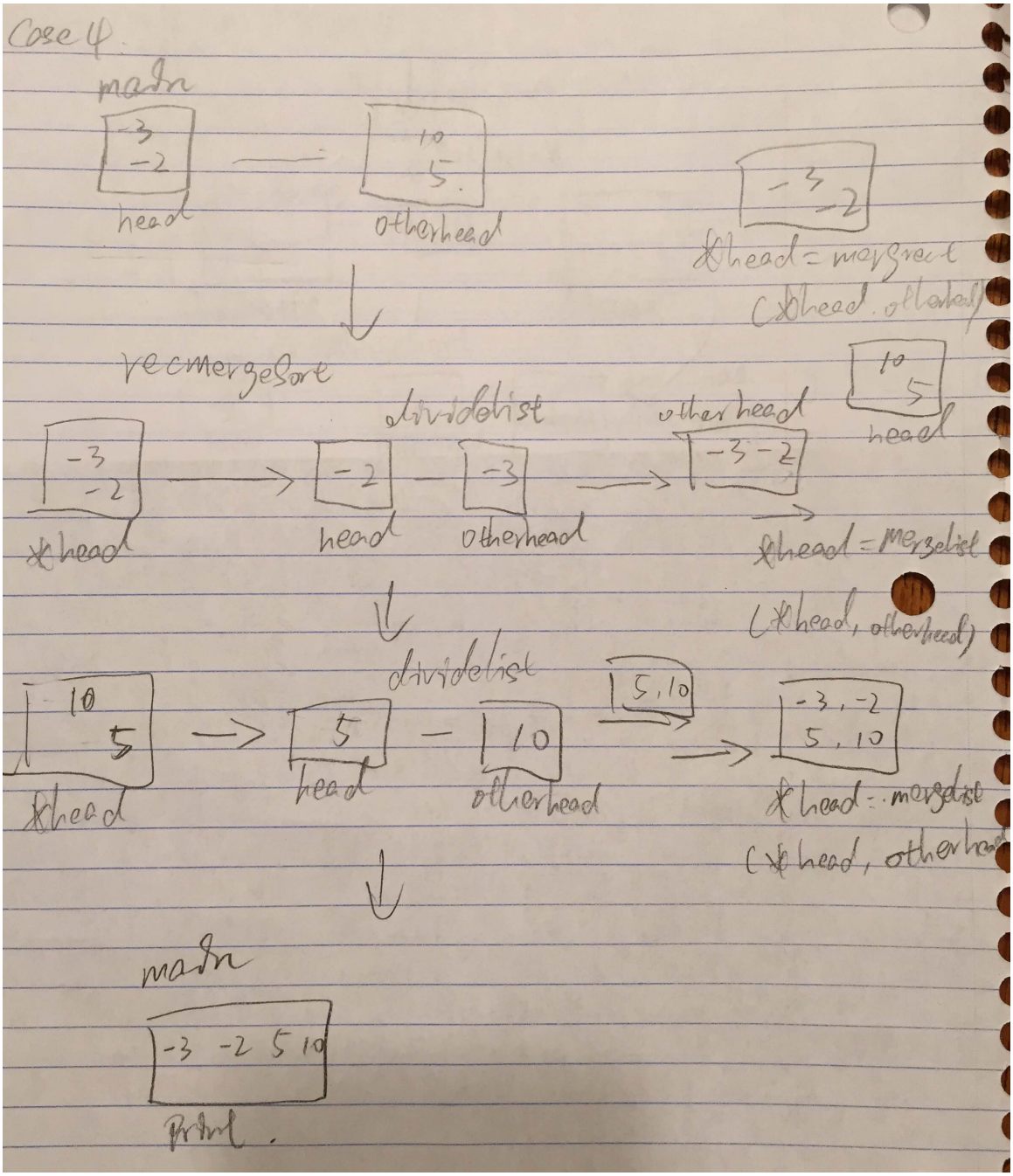
Test case 3: When the user input 2 random numbers like “2, -2”, it calls the recMergeSort, and check it is not empty and not the only one element. So recMergeSort calls divideList, then the recursion is called, and then the numbers of the list divided by two part, one is -2, and the other is 2, and then merges into sorted list with -2 and 2.

Case 3


main
-2
next
2
next
NULL
node head
rec MergeSort
head
-2
next
2
next
NULL
divideList
-2
middle
NULL
2
current
NULL
recMergeSort
-2
head
2
otherhead
each one has only one elem
so doing nothing.
MergeSort
-2
head
2
otherhead
newhead
first
first->next (NULL)
lastSmall->next
first2
main
-2, 2
prhe

Test case 4: In the last case, if the user want to put 4 numbers into the list, like “10, 5, -3, -2”, first the list of length 4 is written to the length function, and the list get it. After that the list passes to the recMergeSort , due to it is not empty and not contain only one element, so it calls the divideList function, it check it is not empty again, and then it will divide into two groups, one contains 5, 10, another contains -2, 3,then it will merges into 5, 10 and -3 and -2, finally it merges into sorted list with -3

-2 5 10.


Case 4.
main
-3
-2
head
10
5
otherhead
recmergesort
-3
-2
*head
-2
head
-3
otherhead
dividelist
-3
-2
otherhead
10
5
head
*head = mergelist
(*head, otherhead)
10
5
*head
5
head
10
otherhead
dividelist
5, 10
-3, -2
5, 10
*head = mergelist
(*head, otherhead
main
-3 -2 5 10
Print.