Meagin Arrocha

3. a) It would twice as long because of the while loop in the insert function; it has to go through the loop until it is done and that will take longer with twice as many inserts.

b) It would be about the same because it is going to do the same amount of work for a smaller list compared to a list twice as big.

c) It is possible for the performance to take longer with a bigger list because you can potentially be searching through ALL of the nodes for a value that is not even there. You can’t pick a specific index and you have to start from the beginning. Also if the value you’re looking for is at the end, then it would take longer than if it were at the head; Otherwise the performance would be the same.

d) Just like C it is also possible here for performance to take longer with a bigger list because even though you’re starting from the tail, the value you’re looking for to remove might be at the head or not even in the list at all and you would have to go through the entire while loop. It could be beneficial if the value you were looking for is at the tail because then the performance would be the same if you were looking for the value in the tail in a small list or a larger one.

e) (current -> next != NULL) has to come before ( && current -> next -> data != val) because otherwise your program will crash. It needs to look for if null is next first because the next node could be at the end of the list and you don’t want to look for a value in that spot of memory when nothing is there.