Jean Gomez

q10

For question 10 we created a linked list class that contained 4 pointers that were all initialized to zero. In the main function we also initialized pointers head and last to nullptr so that we can pass them to fit our function insert which takes parameters of two pointers and a integer value we choose to insert in the linked list. After our linked list has its values set, we then call function OddEvenList so we can put the odd values in the front and even values in the back. We pass the pointer of the list and check that it’s not null and contains values. When we check the value of the first node, if it is even, we will initialize both lists Evenhead and lastEven to the first node. The same goes for the first odd node. Afterwards once we get to the next even node, our Evenhead list will already be set to the current node so now we will be adding on to the lastEven list. Here we will set the next pointer to equal the current node this way we keep adding onto our even list. We will also be updating the last even node to be equal to the current node we passed in. This way if we add a node to the even list then the last node in the even list is that current node we just added. This applies for both odd and even list and when we check which list, we will be adding the node to we continue to update the current node so we keep moving on in the list. When we break out of the while loop, we check that the Oddhead list has values in it so we can initialize our head list to it. Our oddhead list was also initialized to lastodd list but now we need the lastodd list to point to our Even list so we set our lastodd list to point to our evenhead list which contains our list of all even values. Then we need the even list to point to a nullptr so that it won’t point to one of the original values in the list.