Input node in a sorted linked List
5->7->10->
add 9
5->7->9->10->

Questions to ask:

- Is it already sorted?
- What if it is an empty list?
- If not specified: ask if in increasing or decreasing order.
- Any repeating and how to deal with it

Initial algorithm:

- If the list is empty, add the node and make it the head.
- Add element to the list, start from the head and check if the item is bigger than the current one.
    - If it is larger, go to next element, and repeat
    - If smaller
        - If it is at head, make it the head
        - If its anywhere else make connections

Things to consider:

- What is the complexity of this algorithm, O(n)
- What sort of linked list should this be?
    - Singly Linked List
    - Doubly Linked List
    - Circular Linked List
        - Should only need a Singly linked list because Since I dont need to go back, I dont need a double or circular linked list.
- Do I need just head or do I add tail
    - For this no need for tail because either way its going to be O(n)

C++ code:

```
#include <iostream>

//node struct
struct Node{
	int data;
	struct Node *next;
};

Node* newNode(int data){
	Node* new_node = new Node();
	new_node->data = data;
	new_node->next = nullptr;
	return new_node;
}

void insert_sorted(Node** head_ref, Node* new_node){
	Node * current;

	//check if the head node is the end
	if(*head_ref==nullptr || (*head_ref)->data >= new_node->data){
		new_node->next = *head_ref;
		*head_ref = new_node;
	}
	else {
		current = *head_ref;
		while(current->next != nullptr && current->next->data < new_node->data){
			current = current->next;
		}
		new_node->next = current->next;
		current->next = new_node
	}
}

void print_list(Node* head){
	Node* current = head;
	std::cout<<"head->";
	while(current != nullptr){
		std::cout << current->data << "->";
		current = current->next;
	}
}
```

```
int main(){
	Node *head = nullptr;
	Node* new_node = newNode(5);
	insert_sorted(&head, new_node);
	print_list(head);

	Node *head = nullptr;
	Node* new_node = newNode(10);
	insert_sorted(&head, new_node);
	print_list(head);

	Node *head = nullptr;
	Node* new_node = newNode(7);
	insert_sorted(&head, new_node);
	print_list(head);
}
```