```
class Node<T: Equatable> {
    var item: T
    var next: Node<T>?

    init (item: T){
        self.item = item
    }
}

public class Linked_List<T: Equatable & Hashable & Comparable> {
    private var head: Node<T>?

    public var isEmpty: Bool {
        return head == nil
    }

    public func insert_value (value: T){
        var new_Node = Node(item: value)

        if isEmpty{
            head = new_Node
        }
        else {
            var current = head
            while current?.next != nil{
                current = current?.next
            }
            current?.next = new_Node
        }
    }

    public func display() {
        var current_node = head

        while current_node?.item != nil {
            print (current_node?.item)
            current_node = current_node?.next
        }
    }

    public func remove_duplicate() {
        var distinct_set = Set<T>()
        var current_node = head
        while current_node?.next != nil{
            if let current = current_node {
                distinct_set.insert(current.item)
            }
            current_node = current_node?.next
        }
        print (distinct_set.sorted())
    }
}
```

```
var node = Linked_List<Int>()

node.insert_value(value: 1)
node.insert_value(value: 2)
node.insert_value(value: 1)
node.insert_value(value: 4)
node.insert_value(value: 5)
node.insert_value(value: 6)
node.insert_value(value: 5)
node.insert_value(value: 8)

node.display()
node.remove_duplicate()
```