```swift
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
        }
    }
    // display the linked list
    public func display() {
        var current_node = head

        while current_node?.item != nil {
            print (current_node?.item)
            current_node = current_node?.next
        }
    }
    // remove duplicates from the list
    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())
    }
    //reverse the linked list
    public func reverse(){
        var temp1: Node<T>? = nil
        //var temp2: Node<T>? = nil

        while head != nil{
          let temp2 = head?.next
          head?.next = temp1
          temp1 = head
            head = temp2
        }
        head = temp1

        display()
    }
    //find the linked list is palindrome or not
    public func find_Palindrome()-> Bool{
        var fast = head
        var slow = head

        //to find the center node and the last node
        while fast != nil && slow != nil{
            fast = fast?.next?.next
            slow = slow?.next
        }
        //reverse the second half this is nothing but a reverse function
        var temp1: Node<T>? = nil

        while slow != nil{
            var temp2 = slow?.next
            slow?.next = temp1
            temp1 = slow
            slow = temp2
        }
        //Now go through the lists and find palindrome or note
        var left = head
        var right = temp1
        while right != nil{
            if left?.item != right?.item{
                return false
            }
            left = left?.next
            right = right?.next
        }
        return true
    }
}

var node = Linked_List<Int>()

node.insert_value(value: 1)
node.insert_value(value: 4)
node.insert_value(value: 4)
node.insert_value(value: 1)
node.insert_value(value: 1)
node.insert_value(value: 2)
node.insert_value(value: 4)
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.reverse()
var out = node.find_Palindrome()
if out {
    print ("It is a palindrome")
}else {
    print ("It is not a palindrome")
}
//node.find_Palindrome()
//node.display()
//node.remove_duplicate()
```