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

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

public class LinkedList<T: Equatable> {

    private var head: Node<T>?
    public var isEmpty: Bool {
        return head == nil
    }

    public func isAppend(value: T){
        let newNode = Node(item: value)

        if isEmpty{
            head = newNode
        }else{
            var currentNode = head
            while currentNode?.next != nil{
                currentNode = currentNode?.next
            }
            currentNode?.next = newNode
        }
    }

    public func display(){
        var current = head
        while current?.item != nil{
            print (current?.item)
            current = current?.next
        }
    }
    public func delete_item(value: T){
        var current = head
        var previous: Node<T>?
        if current?.item == value{
            head = head?.next
        }
       /* while let unwrapped_current = current {
            if unwrapped_current.item == value{
                previous?.next = unwrapped_current.next
            }
            previous = current
            current = current?.next
        }*/
        //unwrapping the current to a new variable
        while let unwrapped_current = current{
```

```
            if unwrapped_current.item == value{
                previous?.next = unwrapped_current.next
            }
            previous = current
            current = current?.next
        }
    }
}

var element = LinkedList<Int>()
element.isAppend(value: 5)
element.isAppend(value: 4)
element.isAppend(value: 3)
element.isAppend(value: 2)
element.isAppend(value: 1)
element.display()
element.delete_item(value: 3)
print ("\n After Deleting: \n")
element.display()
element.delete_item(value: 5)
print ("\n After Deleting Head: \n")
element.display()
```