```go
// 今回はコメント書かなくて OK です
package main

import "fmt"

type Node struct {
  data int
  next *Node
}

type LinkedList struct {
  head *Node
}

// 位置を指定して挿入
func (list *LinkedList) Insert(data int, pos int) error {
  newNode := &Node{data: data}

  if pos < 0 {
    return fmt.Errorf("場所がありません: %d", pos)
  }
  if pos == 0 {
    newNode.next = list.head
    list.head = newNode
    return nil
  }

  current := list.head

  for i := 0; i < pos-1; i++ {
    if current == nil || current.next == nil {
      return fmt.Errorf("範囲外")
    }
    current = current.next
  }

  newNode.next = current.next
  current.next = newNode

  return nil
}

// 位置を指定して削除
func (list *LinkedList) Delete(pos int) error {
  if list.head == nil {
    return fmt.Errorf("リストにデータがありません")
  }
```

```
  if pos < 0 {
    return fmt.Errorf("場所がありません: %d", pos)
  }

  if pos == 0 {
    return fmt.Errorf("場所がありません: %d", pos)
  }

  current := list.head
  for i := 0; i < pos-1; i++ {
    if current == nil || current.next == nil {
      return fmt.Errorf("範囲外")
    }
    current = current.next
  }

  if current.next == nil {
    return fmt.Errorf("範囲外")
  }

  // 削除というか、ポインタの変更やねこれは
  current.next = current.next.next
  return nil
}

// リストの全要素を表示
func (list *LinkedList) Display() {
  current := list.head

  for current != nil {
    fmt.Printf("%d -> ", current.data)
    current = current.next
  }
  fmt.Println("nil")
}
func main() {
  list := LinkedList{}
  list.Insert(10, 0)
  list.Insert(11, 0)
  list.Insert(12, 0)
  list.Display()

  list.Delete(2)
  list.Display()
}
```