type Node struct{

data int

next \*Node

}

type Stack struct{

top \*Node

}

func(s \*Stack) DisplayStack() {

if s.top==nil{

fmt.Println("stack underflow")

return

}

curr := s.top

for curr!=nil{

fmt.Println(curr.data)

curr=curr.next

}

}

func (s \*Stack) Push(value int) {

newNode :=&Node{data: value}

if s.top==nil{

s.top=newNode

return

}

newNode.next=s.top

s.top=newNode

}

func(s \*Stack) Pop(){

if s.top==nil{

fmt.Println("stack underflow")

return

}

s.top=s.top.next

}

LeetCode

1. 682 Baseball Game

func calPoints(operations []string) int {

s:=&Stack{}

for i:=0;i<len(operations);i++{

if operations[i]=="C"{

s.Pop()

}else if operations[i]=="D"{

s.Push(s.top.data\*2)

}else if operations[i]=="+"{

s.Push(s.top.data+s.top.next.data)

}else{

value,\_:=strconv.Atoi(operations[i])

s.Push(value)

}

}

result:=s.Sum()

return result

}

type Node struct{

data int

next \*Node

}

type Stack struct{

top \*Node

}

func (s \*Stack) Push(value int){

newNode:=&Node{data:value}

if s.top==nil{

s.top=newNode

return

}

newNode.next=s.top

s.top=newNode

}

func (s \*Stack) Sum() int{

sum:=0

if s.top==nil{

return 0

}

curr:=s.top

for curr!=nil{

sum+=curr.data

curr=curr.next

}

return sum

}

func (s \*Stack) Pop(){

if s.top==nil{

return

}

s.top=s.top.next

}

1. 20 Valid Parantheses

func isValid(s string) bool {

st:=&Stack{}

for i:=0;i<len(s);i++{

if s[i]=='(' || s[i]=='[' || s[i]=='{' {

st.Push(s[i])

}else{

if st.top==nil{

return false

}else if s[i]==')' && st.top.data=='('{

st.Pop()

}else if s[i]==']' && st.top.data=='['{

st.Pop()

}else if s[i]=='}' && st.top.data=='{'{

st.Pop()

}else{

return false

}

}

}

if st.top==nil{

return true

}

return false

}

type Node struct {

data byte

next \*Node

}

type Stack struct {

top \*Node

}

func (s \*Stack) Push(value byte) {

newNode:=&Node{data:value}

newNode.next=s.top

s.top=newNode

}

func (s \*Stack) Pop(){

s.top=s.top.next

}