Linked list

1. Add a node at the end & beginning

type Node struct{

data int

next \*Node

}

type List struct{

head \*Node

tail \*Node

}

func (l \*List) addAtHead(value int) {

newNode:=&Node{data: value}

if l.head==nil{

l.head=newNode

l.tail=newNode

return

}

newNode.next=l.head

l.head=newNode

}

func (l \*List) addAtTail(value int) {

newNode:=&Node{data:value}

if l.head==nil{

l.head=newNode

l.tail=newNode

return

}

l.tail.next=newNode

l.tail=newNode

}

func (l \*List) display(){

if l.head==nil{

fmt.Println("nothing in list")

return

}

curr:=l.head

for curr!=nil{

fmt.Println(curr.data)

curr=curr.next

}

}

func main(){

list :=&List{}

list.addAtHead(3)

list.addAtHead(2)

list.addAtHead(1)

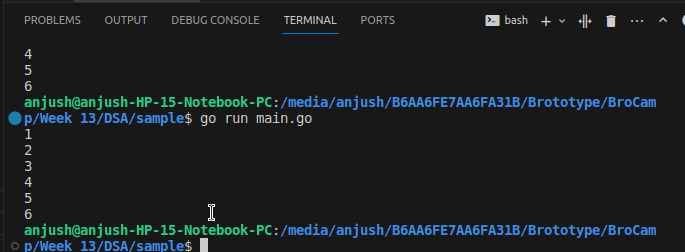
list.addAtTail(4)

list.addAtTail(5)

list.addAtTail(6)

list.display()

}



1. Convert array to a linked list

type Node struct{

data int

next \*Node

}

type List struct{

head \*Node

tail \*Node

}

func (l \*List) display(){

if l.head==nil{

fmt.Println("nothing in list")

return

}

curr:=l.head

for curr!=nil{

fmt.Println(curr.data)

curr=curr.next

}

}

func (l \*List) ConvertArray(arr[] int){

first:=&Node{data: arr[0]}

l.head=first

curr:=l.head

for i:=1;i<len(arr);i++{

newNode:=&Node{data: arr[i]}

curr.next=newNode

curr=curr.next

}

l.tail=curr

}

func main(){

list :=&List{}

arr:=[]int{1,23,4,5,6,7}

list.ConvertArray(arr)

list.display()

}

1. Write a program to remove duplicates in a sorted singly linked list

type Node struct{

data int

next \*Node

}

type List struct{

head \*Node

tail \*Node

}

func (l \*List) display(){

if l.head==nil{

fmt.Println("nothing in list")

return

}

curr:=l.head

for curr!=nil{

fmt.Println(curr.data)

curr=curr.next

}

}

func (l \*List) ConvertArray(arr[] int){

first:=&Node{data: arr[0]}

l.head=first

curr:=l.head

for i:=1;i<len(arr);i++{

newNode:=&Node{data: arr[i]}

curr.next=newNode

curr=curr.next

}

l.tail=curr

}

func (l \*List) RemoveDuplicates(){

if l.head==nil{

fmt.Println("empty list")

return

}

curr:=l.head

follower:=curr.next

for follower!=nil{

if curr.data==follower.data{

curr.next=follower.next

follower=follower.next

}else{

curr=curr.next

follower=follower.next

}

}

curr.next=nil

l.tail=curr

}

func main(){

list :=&List{}

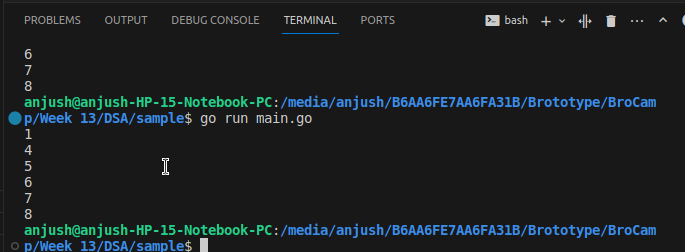
arr:=[]int{1,1,4,5,5,5,5,6,7,8,8}

list.ConvertArray(arr)

list.RemoveDuplicates()

list.display()

}



1. Delete node with the value specified

type Node struct{

data int

next \*Node

}

type List struct{

head \*Node

tail \*Node

}

func (l \*List) display(){

if l.head==nil{

fmt.Println("nothing in list")

return

}

curr:=l.head

for curr!=nil{

fmt.Println(curr.data)

curr=curr.next

}

}

func (l \*List) ConvertArray(arr[] int){

first:=&Node{data: arr[0]}

l.head=first

curr:=l.head

for i:=1;i<len(arr);i++{

newNode:=&Node{data: arr[i]}

curr.next=newNode

curr=curr.next

}

l.tail=curr

}

func (l \*List) Remove(value int) {

if l.head==nil{

fmt.Println("empty list")

return

}

if l.head.data==value{

l.head=l.head.next

return

}

prev:=l.head

curr :=l.head.next

for curr!=nil && curr.data!=value{

prev=curr

curr=curr.next

}

if curr==nil{

fmt.Println("value not in the list")

return

}

prev.next=curr.next

if curr.next==nil{

l.tail=curr.next

}

}

func main(){

list :=&List{}

arr:=[]int{1,1,4,5,6,7,8}

list.ConvertArray(arr)

list.Remove(4)

list.display()

}

