```
######### PROGRAM TO IMPLEMENT DOUBLY LINKED LIST###################
class Node:
    def __init__(self,data):
        self.data=data
        self.next=None
        self.prev=None

class DoublyLinkedList:
    def __init__(self):
        self.head=None
        self.size=0

    def __len__(self):
        return len(self.size)

    def add(self,item):
        newNode = Node(item)
        newNode.next=self.head
        self.head=newNode
        if self.size !=0:
            newNode.next.prev=newNode
        self.size +=1
    def contains(self,target):
        curNode = self.head
        while curNode is not None and curNode.data != target:
            curNode=curNode.next
        return curNode is not None

    def traversal(self):
        curNode = self.head
        while curNode is not None:
            if(curNode.next != None):
                print(curNode.data,end="<->")
            else:
                print(curNode.data,end=".\n")
            curNode=curNode.next

    def RemoveNode(self,target):
        predNode=None
        curNode=self.head
        while curNode is not None and curNode.data != target:
            predNode=curNode
            curNode=curNode.next
        if curNode is not None:
            if curNode is self.head:
                self.head=curNode.next
                curNode.next.prev=None
            else:
                predNode.next=curNode.next
                if curNode.next != None:
                    curNode.next.prev=predNode


Llist=DoublyLinkedList()
```

```
Llist.add(10)
Llist.add(20)
Llist.add(30)
Llist.add(40)
Llist.add(50)
Llist.add(60)
print("Linkled list Data:")
Llist.traversal()
print(Llist.contains(10))
Llist.RemoveNode(10)
print("After Removal")
print("Linkled list Data:")
Llist.traversal()


##########################OUTPUT##########################
'''
Linkled list Data:
60<->50<->40<->30<->20<->10.
True
After Removal
Linkled list Data:
60<->50<->40<->30<->20.

'''
```