Find the sum of the element in double linked list

```
Algorithm sum(L)
    If L. isEmpty return 0
      P:=L.first()
       S:=p.element()
While !L.isLast(p) do
      P:=L.after(p);
      S:=s+p.element()
Return s
```

Level 1

A. Implement in JavaScript the function findMiddle(L) than we did in class. The DLinkedList class is included from the DLinkedList.js file.

Find the middle element in double linked list

```
Algorithm findMiddele(L)
    If L. isEmpty throw Error("empty list").
      f:=L.first();
     e:=L.last()
 while f!=e && L.after(f) !=e  do//!(f==e V L.after(f)==e)
            f:=L.after(f);
            e:=L.before(e)
return e;
```

big o of findeMiddel is O(n)

O(n)

B. Describe, in pseudo-code, how to implement the stack ADT using a DLinkedList. What is the running time of the push() and pop() methods in this case?Implement a new Stack class in JavaScript based on (using) the DLinkedList class like done in A above.

Algorithm push(elem)

insertLast(elem)

big O is O(1)

Algorthim pop()

removeLast()

big O is O(1)

C. Describe, in pseudo-code, how to implement the queue ADT using a DLinkedList. What is the running time of the enqueue() and dequeue() methods in this case? Implement a new Queue class in JavaScript using on the DLinkedList class.

Algorthim enqueue (elem)

insertLast(elem)

big O is O(1)

Algorthim dequeue ()

removeFirst()

big O is O(1)

Level 2

C-2.2 Describe, in pseudo-code, how to implement the queue ADT using two stacks. What is the running time of the enqueue() and dequeue() methods in this case?

Algorthim enqueue (elem)

enueue.push(elem);

```
enqueue(elem) {
    this._enqueue.push(elem)
    }
```

big O of this is O(1)

Algorthim dequeue ()

If !=deque.isEmpty()

Return deque.pop()

Else do

While !enque.isEmpty()

deque.push(enque.pop())

Return deque.pop()

big O is O(n)

```
 dequeue() {
     if(this._dequeue.isEmpty() &&
this._enqueue.isEmpty())
         throw Error("Empty queue")
    if(!this._dequeue.isEmpty())
       return this._dequeue.pop()
    else{
       while(!this._enqueue.isEmpty()){
           this._dequeue.push(this._enqueue.pop())
       }
       return this._dequeue.pop();
    }
    }
```