

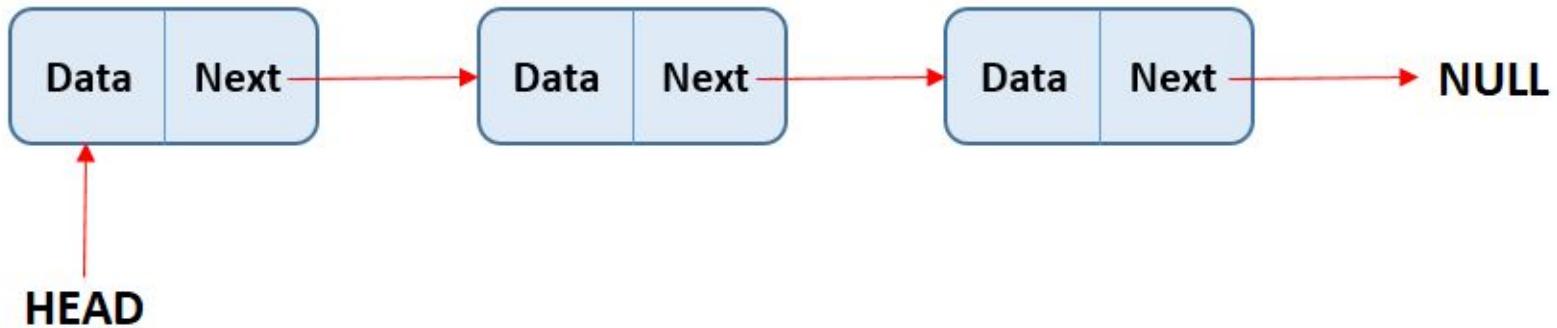
Data Structures

Manage how data is stored and accessed



Linked List

A list where nodes are **linked**, each containing data and a pointer to the **next node**, stored in free memory space.



Node

Nodes hold a **value** and the link
to the **next node**

```
class Node{  
    constructor(data) {  
        this.data = data;  
        this.next = null;  
    }  
}
```

Linked List

Starts with a **head** property
that references the first node.

When we start, the list is
empty, so the head pointer is
null

```
class LinkedList {  
    constructor () {  
        this.head = null;  
    }  
}
```



Demonstration

appendNode



Demonstration

prependNode



Demonstration

pop



Demonstration

removeFromFront



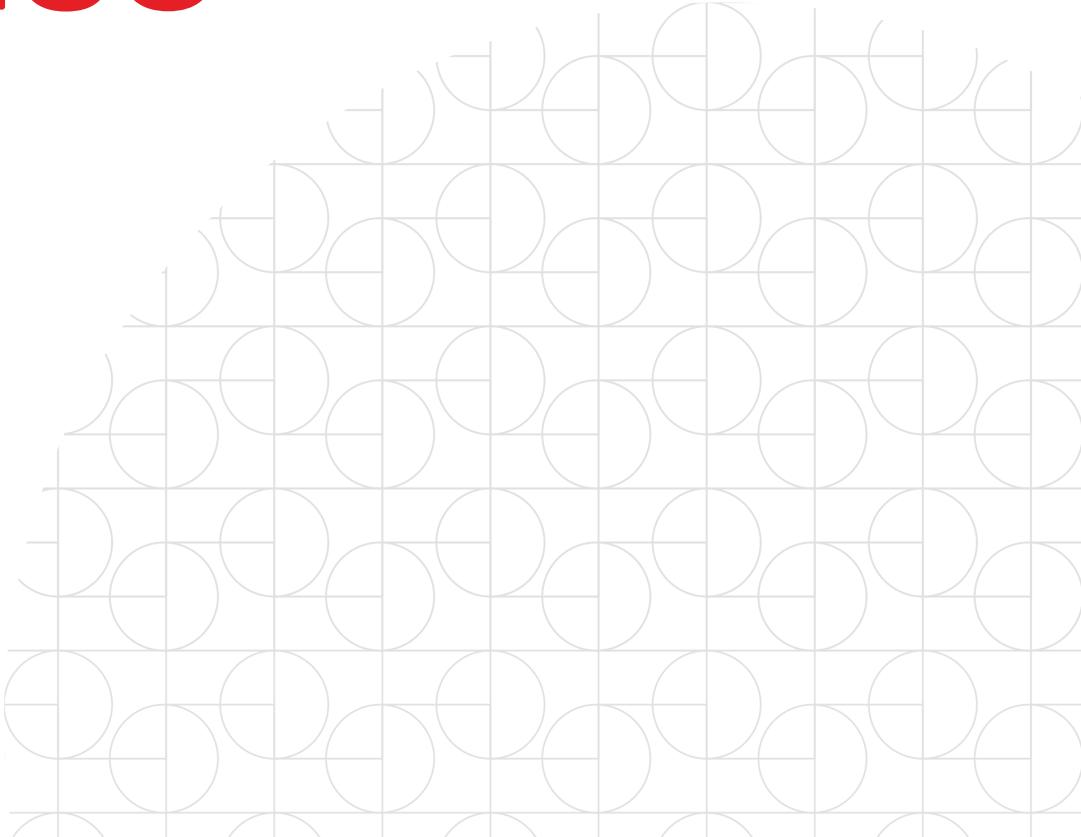
Doubly Linked Lists

A list where nodes are **linked**, each containing data and a pointer to the **next node** and **previous node**.



Examples

- Playlist
- Photo album
- Web browser





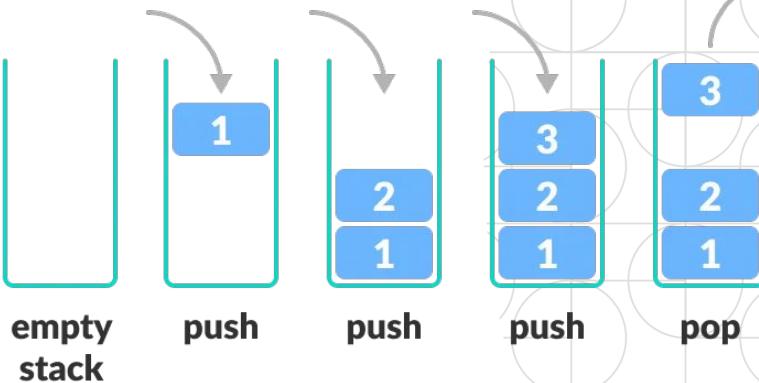
Linked Lists Exercise

Implement the required methods and classes according to the specifications in the comments

<https://git.generalassemb.ly/ENT-SDA-SEB-216-IP/SDA-SIRAJ/tree/main/resources/Computer-Science/11.%20Linked-Lists/exercises>

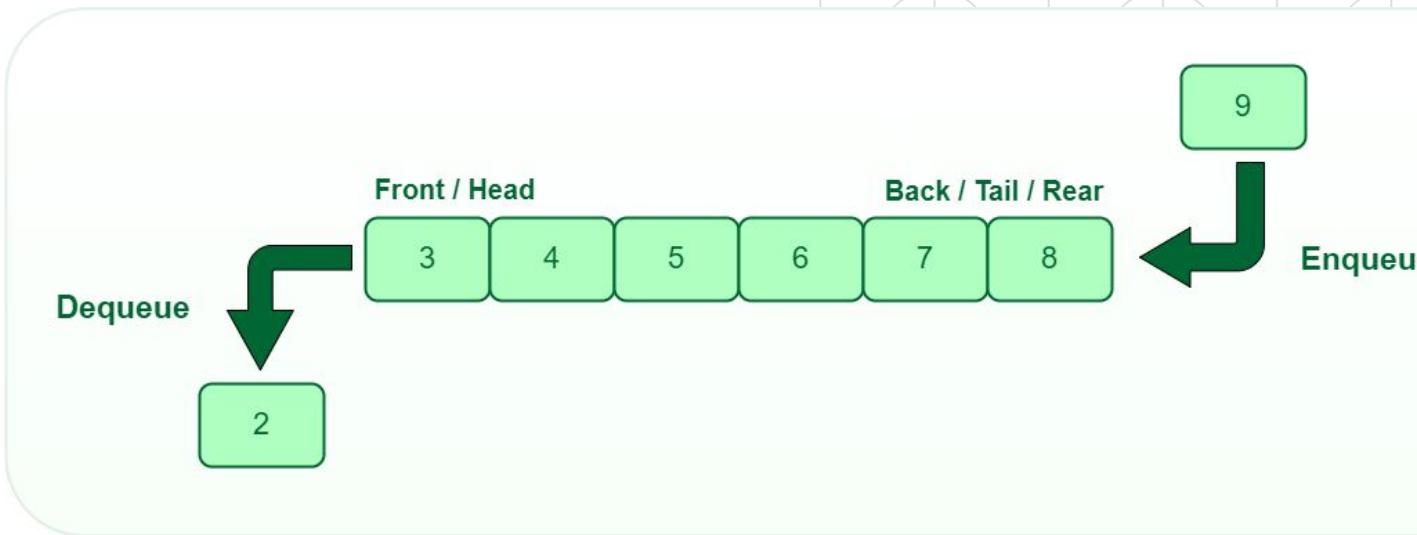
Stacks

Last in, First out behavior. The last, most recently added item, is first to be removed.



Queues

First in, First out behavior. Elements are processed in the order they are added.



Demonstration

Push / Enqueue



Demonstration

Pop / Dequeue



Demonstration

Bracket Matching





Bracket Matching Exercise

Implement the required methods and classes according to the specifications in the comments

<https://git.generalassemb.ly/ENT-SDA-SEB-216-IP/SDA-SIRAJ/tree/main/resources/Computer-Science/12.%20Stacks-An-d-Queues/exercises>