# Lab 4: LinkedList - Stack - Queue

## 1 Singly Linkedlist

Following is a representation of a singly linked list

```
class Node:
def __init__(self, data):
    self.data = data
    self.next = None
```

```
class SingleLinkedList:
def __init__(self, data):
    self.head = None
    self.tail = None
```

Complete the following functions to fulfill the given requirements (linkedlist without `p_tail`):

1. Insert an integer to the head of a given linkedlist:
   - `def addHead(self, head, data)`
2. Insert an integer to the tail of a given linkedlist:
   - `def addTail(self, head, data)`
3. Remove the first `NODE` of a given linkedlist:
   - `def removeHead(self, head)`
4. Remove the last `NODE` of a given linkedlist:
   - `void removeTail(Node* &pHead)`
5. Remove all `NODE` from a given linkedlist:
   - `void removeAll(Node* &pHead)`
6. Remove an integer before a value of a given linkedlist:
   - `void removeBefore(Node* &pHead, int val)`
7. Remove an integer after a value of a given linkedlist:
   - `void romveAfter(Node* &pHead, int val)`
8. Insert an integer at a position of a given linkedlist:
   - `bool addPos(Node* &pHead, int data, int pos)`
9. Remove an integer at a position of a given linkedlist:
   - `void RemovePos(Node* &pHead, int pos)`
10. Insert an integer before a value of a given linkedlist:
    - `void addBefore(Node* &pHead, int data, int val)`
11. Insert an integer after a value of a given linkedlist:
    - `void addAfter(Node* &pHead, int data, int val)`
12. Print all elements of a given linkedlist:
    - `void printList(Node* &pHead)`
13. Count the number of elements linkedlist:
    - `int countElements(Node* &pHead)`
14. Count the number of appearances of a value in a given linkedlist:
    - `int countAppearance(Node* &pHead, int value)`
15. Create a new `List` by reverse a given linkedlist:
    - `Node* reverseList(Node* &pHead)`
16. Remove all duplicates from a given linkedlist:
    - `void removeDuplicate(Node* &pHead)`
17. Remove all `key` value from a given linkedlist:
    - `bool removeElement(Node* &pHead, int key)`

## 2 Doubly Linkedlist

Following is representation of a doubly linked list:

```
class Node:
def __init__(self, data):
    self.data = data
    self.next = None
```

```
class DoublyLinkedList:
def __init__(self):
    self.head = None
    self.prev = None
```

Implement functions to execute the operations from a singly linkedlist section.

## 3 Stack - Queue

Following is the representation of a Singly linked list node:

```
class Node:
def __init__(self, data):
    self.data = data
    self.next = None
```

Utilize the Linked list above, define the data structure of Stack and Queue, then implement functions to execute the following operations:

1. Stack
   - **Initialize** a stack from a given key.
   - **Push** a key into a given stack.
   - **Pop** an element out of a given stack, return the key's value.
   - **Count** the number of elements of a given stack.
   - Determine if a given stack **is empty**.
2. Queue
   - **Initialize** a queue from a given key.
   - **Enqueue** a key into a given queue.
   - **Dequeue** an element out of a given queue, return the key's value.
   - **Count** the number of elements of a given queue.
   - Determine if a given queue **is empty**.