Linked Lists Intro

What is a linked list ?
List created by linking elements called as nodes !!

Simple Pointer

```
int main() {
    int i = 10;
    cout << i << " " << &i << endl;

    int *p = &i;
    cout << p << " " << *p << endl;
}
```

In [ ]:

```
```

Static memory allocation in C++

```
struct Node {
    int data;
};

int main() {
    Node n1;
    n1.data = 10;

    cout << n1.data << endl;

    Node *p1;
    p1 = &n1;

    cout << p1 << " " << (*p1).data << " " << p1->data;
}
```

In [ ]:

```
```

In [ ]:

~~~
```C++
struct Node {
    int data;
};

int main() {
    Node *p1 = new Node; // dynamic memory allocation

    cout << p1 << " " << p1->data;

    delete p1; // deallocation
}
```
~~~

In [ ]:

```

```

In [ ]:

```
struct Node {
    int data;
    Node *next;
};


int main() {
    Node *p1 = new Node; // dynamic memory allocation
    Node *p2 = new Node; // dynamic memory allocation
    p1->data = 10;
    p1->next = NULL;
    p2->data = 20;
    p2->next = NULL;

    cout << p1 << " " << p1->data << " " << p1->next << endl;
    cout << p2 << " " << p2->data << " " << p1->next << endl;

    delete p1; // deallocation
}
```

In [ ]:

```

```

In [ ]:

~~~
```
struct Node {
    int data;
    Node *next;
};

Node* createNode(int data) {
    Node* p = new Node;
    p->next = NULL;
    p->data = data;
    return p;
}

void printNode(Node* n) {
    cout << "data:" << n->data << " ptr:" << n->next << endl;
}

int main() {
    Node *n1 = createNode(10); // dynamic memory allocation
    Node *n2 = createNode(20); // dynamic memory allocation

    printNode(n1);
    printNode(n2);

}
```
~~~

In [ ]:

Node structure and self referential struct

```
// C++
struct Node {
    int data;
    Node *next;
};

// Decimal 0-9  - 10
// Hexa 0-9 ABCDEF - 16
```

Dynamic memory allocation

```
int main() {
    Node *n1 = new Node();
    n1->data = 10;
    n1->next = NULL;

    Node *n2 = new Node();
    n2->data = 20;
    n2->next = n1;
}
```

In [ ]:

```

```

JAVA

```
class Node {
    int data;
    Node next;

    public Node(int data, Node next) {
        this.data = data;
        this.next = next;
    }
}
Node n1 = new Node(10, null);
Node n2 = new Node(20, n1);
```

Python

```
class Node:
    def __init__(self, data: int, next: Node):
        self.data = data
        self.next = next
    def __str__(self):
        print("data", self.data, "next", self.next)

n1 = Node(10, None)
n2 = Node(20, n1)
```

In [ ]:

```

```

In [8]:

```python
class Node:
    def __init__(self, data: int, next):
        self.data = data
        self.next = next
    def __str__(self):
        return f"(data:{self.data} next: {self.next})"

n1 = Node(10, None)
n1 = Node(20, n1)
print(n1)
print(n1.next)
```

```
(data:20 next: (data:10 next: None))
(data:10 next: None)
```

In [ ]:

```

```

Linked List fundamental

```cpp
struct Node {
    int data;
    Node *next;
};

Node* createNode(int data, Node* next) {
    Node* p = new Node;
    p->next = next;
    p->data = data;
    return p;
}

void printNode(Node* n) {
    cout << "data:" << n->data << " next:" << n->next << endl;
}

int main() {
    Node *n1 = createNode(10, NULL); // dynamic memory allocation
    n1 = createNode(20, n1);
    n1 = createNode(30, n1);

    printNode(n1);
    printNode(n1->next);
    printNode(n1->next->next);
}
```

In [ ]:

```

```

~~~
Insert
```C++
~~~

~~~
struct Node {
    int data;
    Node *next;
};

class LinkedList {
    Node *head; // data member

    Node* createNode(int data, Node* next) {
        Node* p = new Node;
        p->next = next;
        p->data = data;
        return p;
    }
    public:
        LinkedList() {this->head = NULL;} // Constructor

        void insertFront(int data) {
            Node * temp = createNode(data, NULL)
            if (head == NULL) {
                head = temp;
            } else {
                temp->next = head;
                head = temp;
            }
        }

        void print() {

            cout << endl;
            Node * temp = head;
            if (head == NULL) cout << "list is empty" << endl;
            while(temp != NULL) {
                cout << "data:" << temp->data << " next:" << temp->next << endl;
                temp = temp->next;
            }
        }
};

int main() {
    LinkedList l1;
    l1.print();

    l1.insertFront(10);
    l1.print();

    l1.insertFront(20);
    l1.print();
}
```
~~~

In [ ]:

Linked List implementation

```
struct Node {
    int data;
    Node *next;
};

class LinkedList {
    Node *head; // data member
    Node *tail;

    Node* createNode(int data, Node* next) {
        Node* p = new Node;
        p->next = next;
        p->data = data;
        return p;
    }
    public:
        LinkedList() {this->head = this->tail = NULL;} // Constructor

        void insertFront(int data) {
            Node * temp = createNode(data, NULL)
            if (head == NULL) {
                head = temp;
                tail = temp;
            } else {
                temp->next = head;
                head = temp;
            }
        }

        void insertEnd(int data) {
            Node * temp = createNode(data, NULL)
            if (head == NULL) {
                head = tail = temp;
            } else {
                tail->next = temp;
                tail = temp;
            }
        }

        void deleteFront() {
            if (head == NULL) {
                cout << "Underflow" << endl;
            } else {
                Node *temp = head;
                head = head->next;
                delete temp;// temp.next = null
            }
        }
        // Delete by address from middle (doesn't work for first and last node)
        void deleteMid(Node *ptr) {
            Node *temp = ptr->next;
            ptr->data = ptr->next->data;
            ptr->next = ptr->next->next;
            delete temp;
        }

        // delete by value is still O(n)

        // iterate and print
        void print() {

            cout << endl;
            Node * temp = head;
            if (head == NULL) cout << "list is empty" << endl;
            while(temp != NULL) {
                cout << "data:" << temp->data << " next:" << temp->next << endl;
                temp = temp->next;
            }
        }
};

int main() {
    LinkedList l1;
    l1.print();

    l1.insertFront(10);
    l1.print();

    l1.insertFront(20);
    l1.print();
}
```

In [ ]:

```

```

In [ ]:

```
### LL vs Array

```

In [ ]:

```
https://leetcode.com/problems/delete-node-in-a-linked-list/
```