LINKED LISTS

☆ A linked list is a special type of data structure similar to arrays but with special properties

An array is stored in a computer's memory at contigous locations. A linked list stores elements in the heap memory not in contigous locations, but at random locations helping us make it dynamic. To traverse across the linked list, we make sure the position of next element is somehow linked to the previous one. Hence the word 'Linked List'. Starting of the linked list is called as its head and its end is called as pointer.

Stacks and Queues utilize a linked list as its foundation. Each element in a linked list is called as a Node. For C++

```
struct Node {
    int data;
    Node * next;
    Node (data1, next) {
        data = data1
```

```
        next = next 1
    }
}
```

Clearly, a Node will store its data and the memory location which points to the next node.

To initialize :

```
Node * y = new Node (element, nullptr)
```

We create a node element in which the next element refers to nullpointer (as our element is at the tail).

Based on this similar logic, we can convert an array to a linked. Code :

```
convertArrayToLL (arr, N) {
    Node * head = new Node (arr[0]);
    Node * mover = head ;
    for (i = 1 → N - 1) {
        Node * temp = new Node (arr[i]);
        mover → next = temp ;
        mover = temp ;
    }
} return head ;
```

Here, clearly, we initialize the head, iterate through the entire

array with our mover, changing its next and moving forward.

We also need to traverse through this linked list.

```
Node* header = convertFromArray(arr, 3);
Node* mover = header;
while (mover != nullptr) {
    cout << mover -> data;
    mover = mover -> next;
    cout << " ";
}
```

We initialize a mover Node which starts at head but moves to its next at each iteration