DSA SERIES

- Learn Coding

Topic to be Covered today

Linked List

LETS START TODAY'S LECTURE

Finding the length of the linked list

```
int getLength(){
    Node* temp = head ;
    int count = 0;

    while(temp!=NULL){
        count++;
        temp=temp->next;
    }
    return count;
}
```

Finding middle node of the linked list

1. Normal method

2. Hare and tortoise algorithm

Normal Method :

```
int getMiddle(){
    int length = getLength();

    int mid = length/2;

    Node* temp = head;

    while(mid--){
        temp = temp->next;
    }

    return temp->data;
}
```

Hare and tortoise algorithm :

```
int hareAndTotorise (){
    Node* slow = head;
    Node* fast = head;

    while(fast != NULL && fast-> next !=NULL){
        slow = slow->next;
        fast = fast->next->next;
    }

    return slow->data;
}
```

Reverse the linked list

```
void reverse(){
    Node* prev = NULL;
    Node* curr = head;
    Node* forward;
    while(curr!=NULL){
        forward = curr->next;
        curr->next = prev;
        prev = curr;
        curr = forward;
    }

    head = prev;
}
```

Delete middle node of the linked list

```
void Delete(){
    if(head == NULL || head->next==NULL){
        return;
    }

    Node* prev = NULL;
    Node* slow =head;
    Node* fast = head;

    while(fast!= NULL && fast->next != NULL){
        fast =fast ->next->next;

        prev = slow;
        slow = slow->next;
    }

    prev->next = slow->next;
    delete slow;

}
```

Remove duplicates from the sorted linked list

```
Node* removeDuplicates(Node* head) {

        Node *curr = head;

        while(curr!=NULL && curr->next !=NULL){
            if(curr->data == curr->next->data){
                Node* duplicate = curr->next;
                curr->next = curr->next->next;

                delete duplicate;
            } else{
                curr = curr->next;
            }
        }

        return head;

}
```

Learn coding

THANK YOU