19/2/24

Leetcode 1

Delete the Middle Node of a Linked List

```
struct ListNode* deleteMiddle(struct ListNode* head){
    struct ListNode* ptr = head;
    struct ListNode* preptr = NULL;
    int count = 0;
    if (head == NULL) {
        return head;
    }
    else if (head->next == NULL) {
        free(head);
        return NULL;
    }
    else {
        while(ptr != NULL){
            count++;
            ptr = ptr->next;
        }
        int n = 0;
        ptr = head;
        while (n != count/2) {
            preptr = ptr;
            ptr = ptr->next;
            n++;
        }
        preptr->next = ptr->next;
        free(ptr);
    }
```

8) WAP (a) TO construct BST
(b) Traverse using inorder, postorder, preorder
(c) Display elements in the tree

```
typedef struct BST
{
    int data ;
    struct BST *left ;
    struct BST *right ;
}node ;
node *create()
{
    node *temp ;
    printf("\nEnter data :") ;
    temp = (node *)malloc(sizeof(node)) ;
    scanf("%d", &temp->data);
    temp->left = temp->right = NULL ;
    return temp ;
}
void insert(node *root, node *temp)
{
    if (temp->data < root->data)
    {
        if(root->left != NULL)
            insert(root->left, temp);
        else
            root->left = temp ;
    }
```

```
    if (temp->data > root->data)
    {
        if (root->right != NULL)
            insert(root->right, temp);
        else
            root->right = temp;
    }
}
void inorder(node *root)
{
    if(root != NULL)
    {
        inorder(root->left);
        printf("%d", root->data);
        inorder(root->right);
    }
}
void postorder(node *root)
{
    if(root != NULL)
    {
        postorder(root->left);
        postorder(root->right);
        printf("%d", root->data);
    }
}
```

```
void preorder (node *root)
{
    if (root != NULL)
    {
        printf ("%d", root->data);
        preorder (root->left);
        preorder (root->right);
    }
}

void main ()
{
    root = NULL;
    root = insert (root, 10);
    root = insert (root, 20);
    root = insert (root, 30);
    root = insert (root, 25);
    root = insert (root, 8);
    root = insert (root, 12);
    root = insert (root, 16);
    printf ("Insertion successful\n");
    inorder (root);
    printf ("\n");
    preorder (root);
    printf ("\n");
    postorder (root);
    printf ("\n");
}
```

Output

Insertion succesful

8 10 12 16 20 25 30

10 8 20 12 16 30 25

8 16 12 25 30 20 10

20.02.24