while (choice != 6);
return 0;
}

Op:

(1) Insert (2) Inorder (3) Preorder (4) Postorder
(5) Display (6) Exit.
Enter your choice : 1.
Enter value to insert : 50
Enter value to insert : 20
Enter value to insert : 60
Enter value to insert 20
Enter value to insert 70
Enter value to insert 30

2

Inorder Traversal: 20 30 50 60 70.

3

Preorder Traversal : 30 20 70 60 50

4

Postorder Traversal : 20 30 60 70 50.

6

Exit.

Leet code:

(I) Odd even linked list.

```
struct ListNode* oddEvenList(struct ListNode* head){
    if(head==NULL||head->next==NULL){
        return head;
    }
    struct ListNode* odd=head;
    struct ListNode* even=head->next;
    struct ListNode* evenHead=even;
```

```
while (even != NULL && even->next != NULL) {
    odd->next = even->next;
    odd = odd->next;
    even->next = odd->next;
    even = even->next; }
odd->next = evenHead;
return head; }
```

(II) Delete the middle node of a linked list.

```
struct ListNode* deleteMiddle (struct ListNode* head){
    if (head == NULL) return NULL;
    struct ListNode* prev = (struct ListNode*) malloc
                            (sizeof(struct ListNode));
    prev->val = 0;
    prev->next = head;
    struct ListNode* preptr = prev;
    struct ListNode* ptr = head;
    while (ptr != NULL && ptr->next != NULL) {
        preptr = preptr->next;
        ptr = ptr->next->next; }
    struct ListNode* temp = preptr->next;
    preptr->next = preptr->next->next;
    free (temp);
    struct ListNode* newHead = prev->next;
    free (prev);
    return newHead; }
```

LAB-7

Date
Page

17-7-24

Q) Write a program
(a) To construct Binary search Tree
(b) Traverse the tree using inorder, postorder, preorder.
(c) Display the elements in the tree.

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(
                                                struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

struct Node* insert(struct Node* root, int value) {
    if (root == NULL) {
        return createNode(value); }
    if (value < root->data) {
        root->left = insert(root->left, value); }
    else if (value > root->data) {
        root->right = insert(root->right, value); }
    return root;
}
```

NJ
17/7/24

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

void postorder (struct Node* root) {
    if (root != NULL) {
        post order (root -> left);
        post order (root -> right);
        printf ("%d", root -> data);
    }
}

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

void display (struct Node *root) {
    printf ("elements in the tree :: ");
    inorder (root);
    printf ("\n");
}
```

```
int main() {
    struct node* root = NULL;
    int choice, value;
do {
    printf("1. Insert \n 2. Inorder traversal \n
        3. Postorder Traversal \n 4. Preorder traversal
        \n 5. display \n 6. Exit \n");
    printf("Enter your choice: ");
    switch (choice) {
        case 1:
            printf("Enter value to insert : ");
            scanf("%d", &value);
            root = insert(root, value);
            break;
        case 2:
            printf("Inorder Traversal : ");
            inorder(root);
            printf("\n"); break;
        case 3:
            printf("Post order Traversal : ");
            postorder(root); printf("\n");
            break;
        case 4:
            printf("Preorder Traversal : ");
            preorder(root); printf("\n");
            break;
        case 5: display(root); break;
        case 6: printf("Exiting... \n");
            break;
        default: printf("Invalid choice! Please enter a
            valid option. \n"); } }
```

Odd Even Linked List - LeetCod

https://leetcode.com/problems/odd-even-linked-list/description/

Relaunch to update

Problem List | Run | Submit | Premium

Description | Editorial | Solutions | Submissions

328. Odd Even Linked List

Solved

Medium | Topics | Companies

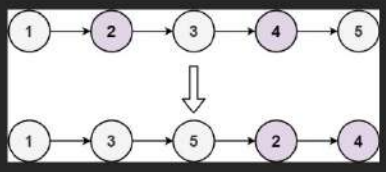
Given the `head` of a singly linked list, group all the nodes with odd indices together followed by the nodes with even indices, and return *the reordered list*.

The **first** node is considered **odd**, and the **second** node is **even**, and so on.

Note that the relative order inside both the even and odd groups should remain as it was in the input.

You must solve the problem in `O(1)` extra space complexity and `O(n)` time complexity.

Example 1:



Input: head = [1,2,3,4,5]

9.6K | 106

Code

C | Auto

```
9    if(head==NULL || head->next==NULL){
10       return head;
11   }
12   struct  ListNode* odd=head;
13   struct  ListNode* even=head->next;
14   struct  ListNode* evenHead=even;
15
16   while(even!=NULL && even->next!=NULL){
17       odd->next=even->next;
```

Saved to local

Ln 13, Col 12

Testcase | Test Result

Accepted Runtime: 2 ms

• Case 1 • Case 2

Input

head =
[1,2,3,4,5]

Output

[1,3,5,2,4]

Delete the Middle Node of a Li

https://leetcode.com/problems/delete-the-middle-node-of-a-linked-list/description/

Relaunch to update

Problem List | Run | Submit | Premium

Description | Editorial | Solutions | Submissions

2095. Delete the Middle Node of a Linked List

Solved

Medium | Topics | Companies | Hint

You are given the `head` of a linked list. **Delete** the **middle node**, and return *the* `head` *of the modified linked list.*

The **middle node** of a linked list of size `n` is the `⌊n / 2⌋th` node from the **start** using **0-based indexing**, where `⌊x⌋` denotes the largest integer less than or equal to `x`.

- For `n` = `1`, `2`, `3`, `4`, and `5`, the middle nodes are `0`, `1`, `1`, `2`, and `2`, respectively.

Example 1:

Input: head = [1,3,4,7,1,2,6]
Output: [1,3,4,1,2,6]
Explanation:
The above figure represents the given linked list. The indices of

3.9K | 30

Code

C | Auto

```c

    struct node {
     int val;
      struct node *next;
   };

struct node* deleteMiddle(struct node* head) {
    if(head==NULL) return NULL;
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
```

Saved to local

Ln 24, Col 17

Testcase | Test Result

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

head =
[1,3,4,7,1,2,6]

Output

[1,3,4,1,2,6]