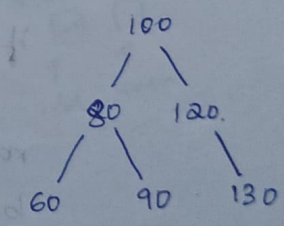
LAB-07 19/02/24

1. BST traversals

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};
struct Node* newNode(int data) {
    struct Node* new = (struct Node*) malloc(sizeof(struct Node));
    new->data = data;
    new->left = NULL;
    new->right = NULL;
    return new;
}
struct Node* insert(struct Node* node, int data) {
    if (node == NULL)
        return newNode(data);
    if (data < node->data)
        node->left = insert(node->left, data);
    else if (data > node->data)
        node->right = insert(node->right, data);
    return node;
}
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) {
        postorder(root->left);
        postorder(root->right);
        printf("%d", root->data);
    }
}
```

19/2/2024.



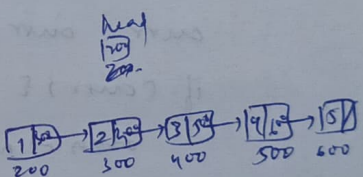
100 80 60 90 120 130

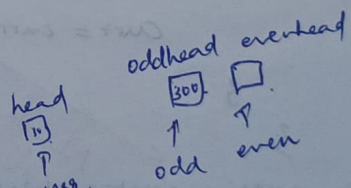
```
void preorder(struct Node* root) {
    if (root!=NULL) {
        printf("%d", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
void display(struct Node* root) {
    if (root!=NULL) {
        display(root->left);
        printf("%d", root->data);
        display(root->right);
    }
}
int main() {
    struct Node* root = NULL;
    int choice, data;
    printf("\n Binary Search Tree Traversal Menu\n");
    printf(" 1. Insert 2. Inorder 3. Preorder 4. Postorder 5. Display 0. Exit\n");
    while(1) {
        printf("Enter your choice: ");
        scanf("%d", &choice);
    switch(choice) {
        case 1: printf("Enter a no. to insert (0 to exit): ");
                scanf("%d", &data);
                if (data==0)
                    break;
                root = insert(root, data);
                break;
        case 2: printf("Inorder: \n");
                inorder(root);
                break;
        Case 3: printf("Preorder: \n");
                preorder(root);
                break;
        Case 4: printf("Postorder: \n");
                postorder(root);
```

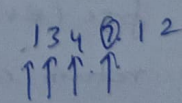
```
        case 5: display(root);
                break;
        case 0: return 0;
        default: printf("Invalid choice\n");
    }
}
return 0;
}
```

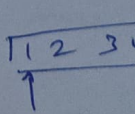
Output:

```
---:- Binary Search Tree Traversal Menu -----
1) Insert 2) Inorder 3) Preorder 4) Postorder 5) Display 0) Exit
Enter your choice: 1
Enter a data to insert: 100
Enter your choice: 1
Enter a data to insert: 80
Enter your choice: 1
Enter a data to insert: 120
Enter your choice: 1
Enter a data to insert: 60
Enter your choice: 1
Enter a data to insert: 90
Enter your choice: 1
Enter a data to insert: 130
Enter your choice: 2
Inorder:
60 80 90 100 120 130
Preorder:
100 80 60 90 120 130
Enter your choice: 4
Postorder:
60 90 80 130 120 100,
Enter your choice: 0
Exiting...
```









Leetcode
19/2/24.

① ②
Exam
19/2/24.

1) Delete middle node:

2) Odd - Even LL:

```
2) struct ListNode* deleteMiddle(struct ListNode* head) {
    if (!head || !head->next) {
        return head;
    }
    struct ListNode* oddHead = (struct ListNode*) malloc(sizeof( ));
    struct ListNode* evenHead = (struct ListNode*) malloc( );
    struct ListNode* odd = oddHead
    struct ListNode* even = evenHead
    struct ListNode* curr = head;
    while (curr) {
        odd->next = curr;
        odd = odd->next;
        curr = curr->next;
        if (curr) {
            even->next = curr;
            even = even->next;
            curr = curr->next;
        }
    }
    even->next = NULL;
    odd->next = evenHead->next;
    struct ListNode* result = oddHead->next;
    free(oddHead);
    free(evenHead);
    return result;
}
```

odd Even List

NULL 19/2/24

29 ~~Odd Even LL~~

1) struct ListNode* deleteMiddle (struct ListNode* head) {

```
    if (!head || (head->next)) {
        return NULL;
    }
    int count=0;
    struct ListNode* curr = head;
    while (curr) {
        count++;
        curr = curr->next;
    }
    int middle = count/2;
    curr = head;
    if (middle == 0) {
        head = head->next;
        free (curr)
        return head;
    }
    for (int i=0; i< middle-1; i++) {
        curr = curr->next;
    }
    struct ListNode* temp = curr->next;
    curr->next = curr->next->next;
    free (temp);
    return head;
}
```

9/2/24