19/2/24

Lab - 7

1) Write a program
a) to construct binary search tree
b) traverse the tree using inorder, postorder, preorder
c) Display elements in the tree

ans)
```
#include <stdio.h>
#include <stdlib.h>
struct treenode
{
    int val;
    struct treenode * left;
    struct treenode * right;
};
struct treenode * createnode (int value)
{
    struct treenode* newnode = (struct treenode*)
                malloc(sizeof (struct treenode));
    newnode -> val = value;
    newnode -> left = NULL;
    newnode -> right = NULL;
    return newnode;
}
struct treenode* insert(struct treenode* node, int value)
{
    if (node == NULL)
    {
        return createnode(value);
    }
    if (value < node -> val)
    {
```

```
        node->left = insert(node->left, value);
    }
    else if (value > node->val)
    {
        node->right = insert(node->right, value);
    }
    return node;
}
void inorder(struct treenode* node)
{
    if (node != NULL)
    {
        inorder(node->left);
        printf("%d", node->val);
        inorder(node->right);
    }
}
void postorder(struct treenode* node)
{
    if (node != NULL)
    {
        postorder(node->left);
        postorder(node->right);
        printf("%d", node->val);
    }
}
void preorder(struct treenode* node)
{
    if (node != NULL)
    {
        printf("%d", node->val);
        preorder(node->left);
        preorder(node->right);
    }
}
```

```
void main()
{
    struct treenode* root = NULL;
    int n, value, choice;
    printf("Menu\n 1. Create tree\n 2. Preorder\n
    3. Inorder\n 4. Postorder\n 5. Exit");
    while(1)
    {
        printf("Enter your choice:");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:
                printf("Enter the no. of nodes:");
                scanf("%d", &n);
                printf("Enter the value of the
                nodes:");
                for(int i=0; i<n; i++)
                {
                    scanf("%d", &value);
                    root = insert(root, value);
                }
                break;
            case 2:
                printf("Preorder traversal:");
                preorder(root);
                printf("\n");
                break;
            case 3:
                printf("Inorder traversal:");
                inorder(root);
                printf("\n");
                break;
            case 4:
                printf("Postorder traversal:");
```

```
            postorder(root);
            printf("\n");
            break;
        case 5:
            exit(0);
            break;
        default:
            printf("Invalid input");
    }
  }
}
```

OUTPUT:

Menu

1. Create tree
2. Preorder
3. Inorder
4. Postorder
5. Exit

Enter your choice: 1

Enter the number of nodes: 6

Enter the values of the nodes: 2 5 7 8 6 9

Enter your choice: 2

Preorder traversal: 2 5 7 6 8 9

Enter your choice: 3

Inorder traversal: 2 5 6 7 8 9

Enter your choice: 4

Postorder Traversal: 6 9 8 7 5 2

Enter your choice: 5

```
MENU:
1. Create tree
2. Preorder
3. Inorder
4. Postorder
5. Exit

Enter your choice: 1
Enter the number of nodes: 6
Enter the values of the nodes: 2 5 7 8 6 9

Enter your choice: 2

Inorder traversal: 2 5 6 7 8 9

Enter your choice: 3

Inorder traversal: 2 5 6 7 8 9

Enter your choice: 4

Postorder traversal: 6 9 8 7 5 2

Enter your choice: 5
```

classmate
Date
Page

1) 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

ans)

```
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int val;
    struct node *next;
};
struct node *head=NULL, *newnode, *temp;
void createlist()
{
    int i, n;
    printf("Enter the no. of nodes:");
    scanf("%d", &n);
    printf("Enter the elements:");
    for(i=0; i<n; i++)
    {
        newnode=(struct node*) malloc (sizeof(struct node));
        scanf("%d", &newnode->val);
        newnode->next=NULL;
        if(head==NULL)
        {
            temp=head=newnode;
        }
        else
        {
            temp->next=newnode;
            temp=newnode;
        }
    }
}
```

```
struct node* oddevenlist(struct node* head)
{
    if (head == NULL || head->next == NULL || head->next->next == NULL)
        return head;
    struct node* oddhead = head;
    struct node* evenhead = head->next;
    struct node *oddcurrent = oddhead;
    struct node *evencurrent = evenhead;
    while (evencurrent != NULL && evencurrent->next != NULL)
    {
        oddcurrent->next = evencurrent->next;
        oddcurrent = oddcurrent->next;
        evencurrent->next = oddcurrent->next;
        evencurrent = evencurrent->next;
    }
    oddcurrent->next = evenhead;
    return oddhead;
}
void display(struct node* head)
{
    struct node* current = head;
    while (current != NULL)
    {
        printf("%d", current->val);
        current = current->next;
    }
    printf("\n");
}
void main()
{
    printf("create linked list:");
    createlist();
    printf("original linked list:");
    display(head);
```

```
        head = oddevenlist(head);
        printf("Linked list after reordering odd and
        even indices:");
        display(head);
    }
}
```

OUTPUT:

```
Create linked list:
Enter the number of elements: 5
Enter the elements: 1 2 3 4 5
original linked list: 1 2 3 4 5
Linked list after reordering odd and even
indices: 1 3 5 2 4
```

</> Code

C Auto

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* oddEvenList(struct ListNode* head) {
    if (head == NULL || head->next == NULL || head->next->next == NULL)
        return head;

    struct ListNode *oddHead = head;
    struct ListNode *evenHead = head->next;
    struct ListNode *oddCurrent = oddHead;
    struct ListNode *evenCurrent = evenHead;

    while (evenCurrent != NULL && evenCurrent->next != NULL) {
        oddCurrent->next = evenCurrent->next;
        oddCurrent = oddCurrent->next;
        evenCurrent->next = oddCurrent->next;
        evenCurrent = evenCurrent->next;
    }
     oddCurrent->next = evenHead;

    return oddHead;
}
```

Testcase | Test Result

Accepted Runtime: 0 ms

• **Case 1** • Case 2

Input

head =

```
[1,2,3,4,5]
```

Output

```
[1,3,5,2,4]
```

Expected

```
[1,3,5,2,4]
```

3) You are given the head of a linked list. Delete the middle node and return the head of the modified linked list

ans)

```
#include <stdio.h>
#include <stdlib.h>
struct node
{
    int data;
    struct node* next;
};
int countnodes( struct node* temp)
{
    int count = 0;
    while (temp != NULL)
    {
        temp = temp -> next;
        count++;
    }
    return count;
}
struct node* deletemid ( struct node* head)
{
    if (head == NULL || head -> next == NULL)
    {
        return NULL;
    }
    struct node* temp = head;
    int count = countnodes ( head);
    int mid = count / 2;
    while (mid-- > 1)
    {
        temp = temp -> next;
    }
```

```
        temp->next = temp->next->next;
        return head;
}
void display (struct node * temp)
{
    while (temp != NULL)
    {
        printf ("%d ", temp->data);
        temp = temp->next;
    }
    printf ("\n");
}
struct node * create (int data);
{
    struct node* temp = (struct node*) malloc( sizeof
                        ( struct node));
    temp->data = data;
    temp->next = NULL;
    return temp;
}
void main()
{
    int value, n;
    struct node* head = NULL, * current = NULL;
    printf (" Enter no. of nodes:");
    scanf ("%d", &n);
    printf (" Enter the values of the nodes:");
    for (int i=0; i<n; i++)
    {
        scanf ("%d", &value);
        struct node * newnode = create (value);
        if (head = NULL)
        {
            head = newnode;
            current = head;
        }
```

```
        else
        {
            current -> next = newnode;
            current = current -> next;
        }
    }
    printf("Given linked list:\n");
    display(head);
    head = deletemid(head);
    printf("Linked list after deleting middle node:\n");
    display(head);
}
```

OUTPUT:

Enter the no. of nodes: 5

Enter the values of the nodes: 1 2 3 4 5

Given linked list:

1 2 3 4 5

Linked list after deleting middle node:

1 2 4 5

</> Code

C ⌄ 🔒 Auto

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
int countOfNodes(struct ListNode* head)
{
    int count = 0;
    while (head != NULL) {
        head = head->next;
        count++;
    }
    return count;
}
struct ListNode* deleteMiddle(struct ListNode* head)
{
    if (head == NULL)
        return NULL;
    if (head->next == NULL) {
        free(head);
        return NULL;
    }
    struct ListNode* copyHead = head;
    int count = countOfNodes(head);
    int mid = count / 2;
    while(mid-->1)
    head = head->next;
    head->next = head->next->next;
    return copyHead;
}
```

Testcase | Test Result

Accepted Runtime: 2 ms

• **Case 1** • Case 2 • Case 3

Input

head =

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

Output

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

Expected

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