Week-8

Program to

a. Construct a 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 data){
   struct node* newnode =(struct node*)malloc(sizeof(struct node));
   newnode->data = data;
   newnode->left = NULL;
   newnode->right = NULL;
   return newnode;
}
struct node* insert(struct node* root, int data){
        if(root == NULL){
            return createNode(data);
        }else{
            if(data <= root->data){
                root->left = insert(root->left, data);
            }else{
                root->right = insert(root->right, data);
            }
            return root;
        }
}
```

```
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);
    }
}
void preorder(struct node *root){
    if(root!=NULL){
        printf("%d",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}
void display(struct node *root){
    printf("\nInorder traversal:");
    inorder(root);
    printf("\n\n Postorder traversal:");
    postorder(root);
    printf("\n\n preorder traversal:");
    preorder(root);
    printf("\n");
}
```

```
int main()
{
  struct node * root = NULL;
  int data;
  char choice;
  do {
      printf("1. Insert into BST \t 2. Display BST
              \t Exit");
      printf("Enter your choice");
      scanf("%c", &choice);

  switch(choice){
      Case '1':
        printf("Enter data to be inserted:");
        scanf("%d", &data);
        root = insert(root, data);
        break;
      Case '2':
        if(root==NULL)
        { printf("BST is empty");
        }else{
            printf("Elements in the tree");
            display(root);
        }
        break;
      Case '3';
          printf("Exiting program");
          break;
      default:
          printf("Invalid choice");
  }
  while(choice != '3');
  return 0;
  }
```

o/p:

1. Insert into BST
2. Display BST
3. Exit

Enter your choice: 1
Enter data: 2
Enter your choice: 1
Enter data: 5
Enter your choice: 1
Enter data: 7
Enter your choice: 1
Enter data: 4
Enter your choice: 1
Enter data: 6
Enter your choice: 1
Enter data: 1
Enter your choice: 2

Elements in the tree:

Inorder Traversal : 1 2 4 5 6 7
Postorder Traversal : 1 4 6 7 5 2
Preorder Traversal : 2 1 5 4 7 6.

② Deleting the middle node of linked list.

⇒

```
struct ListNode* deleteMiddle(struct ListNode*
                              head)
{   int count=0;
    struct ListNode *ptr;
    ptr=head;
    while(ptr!=NULL)
    {   count++;
        ptr=ptr->next;
```

```
int middle = count/2;
struct ListNode * prev = NULL;
ptr = head;
if(head -> next == NULL)
{ phead = NULL;
  return head;
}
for(int i=0; i<middle; i++)
   {
      prev = ptr;
      ptr = ptr -> next;
   }
prev -> next = ptr -> next;
free(ptr);
return head;
}
```

---

(3) Odd Even Linked List.

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