Week-8

(a) To Implement Inorder, preorder, Postorder Traversal

Code:

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

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

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

```
void postorder(struct node
if(root == NULL) return;
postorder(root->left);
postorder(root->right);
printf("%d ", root->data);
}
void display(struct node *root){
printf("BST elements (inorder): ");
inorder(root);
printf("\n");
}

int main() {
struct node *root = NULL;
int choice, value;
while(1) {
printf("\n--- Binary Search Tree Menu ---");
printf("1. Insert into BST\n");
printf("2. Inorder Traversal\n");
printf("3. Preorder Traversal\n");
printf("4. Postorder Traversal\n");
printf("5. Display BST\n");
printf("6. Exit\n");
printf("Enter choice: ");
scanf("%d", &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("Preorder Traversal");
        preorder(root);
        printf("\n");
        break;
    case 4:
        printf("Post order Traversal: ");
        postorder(root);
        printf("\n");
        break;
    case 5:
        display(root);
        break;
    case 6:
        exit(0);
    default: printf("Invalid choice\n");
}
}
```

```
--- Binary Search tree Menu ---
1. Insert into BST
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter choice: 1
Enter value to insert: 23 45 567

Enter choice: 1
Enter value to insert: 45
Enter choice: 1
Enter value to insert: 567
Enter choice: 2
Inorder Traversal: 23 45 567
Enter choice: 4
Postorder Traversal: 567 45 23
Enter choice: 3
Preorder Traversal: 23 45 567
```

Enter choice : 5
Enter (element) : 22 45 567

Enter choice : 6

Exiting. ...

Leet code : linked list cycle

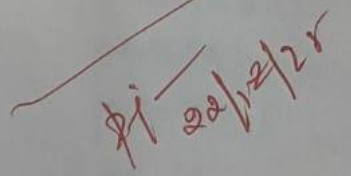
class Solution {
public :

bool has_cycle(ListNode * head) {
if (!head || !head->next) return false;

ListNode * slow = head
ListNode * fast = head
while (fast && fast->next) {
slow = slow->next
fast = fast->next->next;
if (slow == fast) return true;
}.

return false;

};



```c
#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);
    }
    if (data < root->data) {
        root->left = insert(root->left, data);
    } else if (data > root->data) {
        root->right = insert(root->right, data);
    }
    return root;
}

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

Failed

```c
        inorderTraversal(root->right);
    }
}

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

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

void displayBST(struct Node* root) {
    if (root == NULL) {
        printf("BST is empty.\n");
        return;
    }
    printf("Inorder Traversal: ");
    inorderTraversal(root);
    printf("\n");
    printf("Preorder Traversal: ");
    preorderTraversal(root);
    printf("\n");
```

```c
    printf("Postorder Traversal: ");
    postorderTraversal(root);
    printf("\n");
}

int main() {
    struct Node* root = NULL;
    int choice, data;

    while (1) {
        printf("\nBinary Search Tree Operations:\n");
        printf("1. Insert a node\n");
        printf("2. Inorder Traversal\n");
        printf("3. Preorder Traversal\n");
        printf("4. Postorder Traversal\n");
        printf("5. Display BST\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter data to insert: ");
                scanf("%d", &data);
                root = insert(root, data);
                printf("Node inserted successfully.\n");
                break;
            case 2:
                printf("Inorder Traversal: ");
                inorderTraversal(root);
                printf("\n");
                break;
```

```
            case 3:
                printf("Preorder Traversal: ");
                preorderTraversal(root);
                printf("\n");
                break;
            case 4:
                printf("Postorder Traversal: ");
                postorderTraversal(root);
                printf("\n");
                break;
            case 5:
                displayBST(root);
                break;
            case 6:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice. Please try again.\n");
        }
    }

    return 0;
}

```

```
Enter your choice: 1
Enter data to insert: 56
Node inserted successfully.

Binary Search Tree Operations:
1. Insert a node
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter your choice: 1
Enter data to insert: 40
Node inserted successfully.

Binary Search Tree Operations:
1. Insert a node
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter your choice: 5
Inorder Traversal: 3 10 26 40 56
Preorder Traversal: 10 3 26 56 40
Postorder Traversal: 3 40 56 26 10

Binary Search Tree Operations:
1. Insert a node
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter your choice: 6
Exiting...
```

```
Binary Search Tree Operations:
1. Insert a node
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter your choice: 1
Enter data to insert: 10
Node inserted successfully.

Binary Search Tree Operations:
1. Insert a node
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter your choice: 1
Enter data to insert: 26
Node inserted successfully.

Binary Search Tree Operations:
1. Insert a node
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
5. Display BST
6. Exit
Enter your choice: 1
Enter data to insert: 3
Node inserted successfully.

Binary Search Tree Operations:
1. Insert a node
2. Inorder Traversal
3. Preorder Traversal
4. Postorder Traversal
```