

Week-8

② 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 tnode *root) {
    if (root == NULL) return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->data);
}

```

```

void display(struct tnode *root) {
    printf("BST elements (inorder): ");
    inorder(root);
    printf("\n");
}

```

```

int main() {
    struct tnode *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("Postorder traversal: ");
        postorder(root);
        printf("\n");
        break;

```

```

    case 5:
        display(root);
        break;

```

```

    case 6:
        exit(0);
        printf("Exiting...");
}

```

o/p

--- 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: 2  
Enter value to insert: 45

Enter choice: 1  
Enter value to insert: 567

Enter choice: 2  
Inorder traversal: 23 45 567

Enter choice: 3  
Preorder traversal: 567 45 23

Enter choice: 3  
Preorder traversal: 23 45 567

Enter choice: 5  
Rr (element): 22 45 567  
Enter choice: 6  
Exiting...

Leet code: linked list cycle

class Solution {

public:

bool hasCycle(ListNode \*head) {

if (!head || head->next == null) 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;

20/2/21

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct Node {
5      int data;
6      struct Node* left;
7      struct Node* right;
8  };
9
10 struct Node* createNode(int data) {
11     struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
12     newNode->data = data;
13     newNode->left = NULL;
14     newNode->right = NULL;
15     return newNode;
16 }
17
18 struct Node* insert(struct Node* root, int data) {
19     if (root == NULL) {
20         return createNode(data);
21     }
22     if (data < root->data) {
23         root->left = insert(root->left, data);
24     } else if (data > root->data) {
25         root->right = insert(root->right, data);
26     }
27     return root;
28 }
29
30 void inorderTraversal(struct Node* root) {
31     if (root != NULL) {
32         inorderTraversal(root->left);
33         printf("%d ", root->data);

```

⊗ Failed t

```

34     inorderTraversal(root->right);
35 }
36 }
37
38 void preorderTraversal(struct Node* root) {
39     if (root != NULL) {
40         printf("%d ", root->data);
41         preorderTraversal(root->left);
42         preorderTraversal(root->right);
43     }
44 }
45
46 void postorderTraversal(struct Node* root) {
47     if (root != NULL) {
48         postorderTraversal(root->left);
49         postorderTraversal(root->right);
50         printf("%d ", root->data);
51     }
52 }
53
54 void displayBST(struct Node* root) {
55     if (root == NULL) {
56         printf("BST is empty.\n");
57         return;
58     }
59     printf("Inorder Traversal: ");
60     inorderTraversal(root);
61     printf("\n");
62     printf("Preorder Traversal: ");
63     preorderTraversal(root);
64     printf("\n");

```

```

65     printf("Postorder Traversal: ");
66     postorderTraversal(root);
67     printf("\n");
68 }
69
70 int main() {
71     struct Node* root = NULL;
72     int choice, data;
73
74     while (1) {
75         printf("\nBinary Search Tree Operations:\n");
76         printf("1. Insert a node\n");
77         printf("2. Inorder Traversal\n");
78         printf("3. Preorder Traversal\n");
79         printf("4. Postorder Traversal\n");
80         printf("5. Display BST\n");
81         printf("6. Exit\n");
82         printf("Enter your choice: ");
83         scanf("%d", &choice);
84
85         switch (choice) {
86             case 1:
87                 printf("Enter data to insert: ");
88                 scanf("%d", &data);
89                 root = insert(root, data);
90                 printf("Node inserted successfully.\n");
91                 break;
92             case 2:
93                 printf("Inorder Traversal: ");
94                 inorderTraversal(root);
95                 printf("\n");
96                 break;

```

```
97         case 3:
98             printf("Preorder Traversal: ");
99             preorderTraversal(root);
100             printf("\n");
101             break;
102         case 4:
103             printf("Postorder Traversal: ");
104             postorderTraversal(root);
105             printf("\n");
106             break;
107         case 5:
108             displayBST(root);
109             break;
110         case 6:
111             printf("Exiting...\n");
112             exit(0);
113         default:
114             printf("Invalid choice. Please try again.\n");
115     }
116 }
117
118 return 0;
119 }
120
```

```
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