Program 8

Write a program
a) To construct a binary Search tree.
b) To traverse the tree using all the methods i.e., in-order, preorder and post order
c) To display the elements in the tree.

Code:

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

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

```
        preOrder(root->right);
    }
}

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

int main() {
    struct Node* root = NULL;
    int n, value;

    printf("Enter the number of elements to insert in the BST: ");
    scanf("%d", &n);

    printf("Enter %d elements:\n", n);
    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        root = insert(root, value);
    }

    printf("\nIn-order Traversal: ");
    inOrder(root);

    printf("\nPre-order Traversal: ");
    preOrder(root);

    printf("\nPost-order Traversal: ");
    postOrder(root);

    return 0;
}
```

```
Enter the number of elements to insert in the BST: 5
Enter 5 elements:
12 23 45 65 3

In-order Traversal: 3 12 23 45 65
Pre-order Traversal: 12 3 23 45 65
Post-order Traversal: 3 65 45 23 12
```

23/12

Q. WAP a) to construct a Binary Search Tree
b) to traverse the tree using all the methods ie inorder, preorder, postorder
Display all traversal orders

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

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

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

int main() {
    struct Node* root = NULL;
    int n, value;

    printf("Enter the no elements to insert
                in the BST");
    scanf("%d", &n);
    printf("Enter %d elements : \n", n);
    for(int i=0; i<n; i++) {
        scanf("%d", &value);
        root = insert(root, value);
    }

    printf("\n InOrder Traversal: ");
    inOrder(root);

    printf("\n PreOrder Traversal: ");
    preOrder(root);

    printf("\n PostOrder Traversal: ");
    postOrder(root);

    return 0;
}
```

output

```
Enter the number of elements to insert
in the BST : 5

Enter 5 elements : 12 23 45 65 3

InOrder Traversal : 3 12 23 45 65
PreOrder Traversal : 12 3 23 45 65
PostOrder Traversal : 3 65 45 23 12
```