```c
#include <stdio.h>
#include <stdlib.h>

/* Structure of BST node */
struct node {
    int data;
    struct node *left;
    struct node *right;
};

/* Create a new node */
struct node* createNode(int value) {
    struct node *temp = (struct node *)malloc(sizeof(struct node));
    temp->data = value;
    temp->left = NULL;
    temp->right = NULL;
    return temp;
}

/* Insert node into BST */
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
        root->right = insert(root->right, value);

    return root;
}

/* In-order traversal */
void inorder(struct node *root) {
    if (root != NULL) {
        inorder(root->left);
        printf("%d ", root->data);
        inorder(root->right);
    }
}

/* Pre-order traversal */
void preorder(struct node *root) {
    if (root != NULL) {
```

```
        printf("%d ", root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

/* Post-order traversal */
void postorder(struct node *root) {
    if (root != NULL) {
        postorder(root->left);
        postorder(root->right);
        printf("%d ", root->data);
    }
}

/* Main function */
int main() {
    struct node *root = NULL;
    int n, i, value;

    printf("Enter number of nodes: ");
    scanf("%d", &n);

    printf("Enter elements: ");
    for (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;
}
```

Input
50 30 70 20 40 60 80

Output
In-order: 20 30 40 50 60 70 80
Pre-order: 50 30 20 40 70 60 80
Post-order: 20 40 30 60 80 70 50