```
#include <stdio.h>
#include <stdlib.h>
struct tnode {
  int data;
  struct tnode *right, *left;
};
typedef struct tnode TNODE;
TNODE *CreateBST(TNODE *, int);
void Preorder(TNODE *);
void Postorder(TNODE *);
void Inorder(TNODE *);
int main() {
  TNODE *root = NULL;
  int opn, elem, n, i;
  do {
    printf("\nBinary Search Tree Operations\n\n");
    printf("1. Creation of BST\n2. Inorder\n3. Preorder\n4. Postorder\n");
    printf("\nEnter Your Option: ");
    scanf("%d", &opn);
    switch (opn) {
    case 1:
      root = NULL;
      printf("\n\nBST for How Many Nodes? ");
```

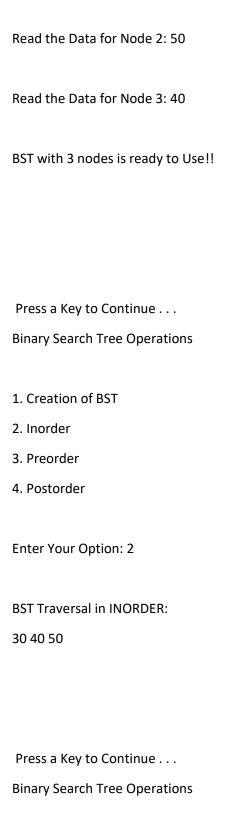
```
scanf("%d", &n);
  for (i = 1; i <= n; i++) {
    printf("\nRead the Data for Node %d: ", i);
    scanf("%d", &elem);
    root = CreateBST(root, elem);
  }
  printf("\nBST with %d nodes is ready to Use!!\n", n);
  break;
case 2:
  printf("\nBST Traversal in INORDER:\n");
  Inorder(root);
  break;
case 3:
  printf("\nBST Traversal in PREORDER:\n");
  Preorder(root);
  break;
case 4:
  printf("\nBST Traversal in POSTORDER:\n");
  Postorder(root);
  break;
default:
  printf("\n\nInvalid Option!!! Try Again!!!\n\n");
  break;
}
printf("\n\n\n Press a Key to Continue . . . ");
```

```
getchar();
  } while (opn != 5);
  return 0;
}
TNODE *CreateBST(TNODE *root, int elem) {
  if (root == NULL) {
    root = (TNODE *)malloc(sizeof(TNODE));
    root->left = root->right = NULL;
    root->data = elem;
    return root;
  } else {
    if (elem < root->data)
       root->left = CreateBST(root->left, elem);
    else if (elem > root->data)
       root->right = CreateBST(root->right, elem);
    else
       printf("Duplicate Element!! Not Allowed!!!");
    return (root);
  }
}
void Preorder(TNODE *root) {
  if (root != NULL) {
    printf("%d ", root->data);
    Preorder(root->left);
    Preorder(root->right);
  }
```

```
}
void Postorder(TNODE *root) {
  if (root != NULL) {
    Postorder(root->left);
    Postorder(root->right);
    printf("%d ", root->data);
  }
}
void Inorder(TNODE *root) {
  if (root != NULL) {
    Inorder(root->left);
    printf("%d ", root->data);
    Inorder(root->right);
  }
}
Output:
Binary Search Tree Operations
1. Creation of BST
2. Inorder
3. Preorder
4. Postorder
```

Enter Your Option: 1

BST for How Many Nodes? 3
Read the Data for Node 1: 50
Read the Data for Node 2: 30
Read the Data for Node 3: 60
BST with 3 nodes is ready to Use!!
Press a Key to Continue
Binary Search Tree Operations
1. Creation of BST
2. Inorder
3. Preorder
4. Postorder
Enter Your Option: 1
BST for How Many Nodes? 3
Read the Data for Node 1: 30



- 1. Creation of BST
- 2. Inorder

3. Preorder
4. Postorder
Enter Your Option: 3
BST Traversal in PREORDER:
30 50 40
Press a Key to Continue
Binary Search Tree Operations
1. Creation of BST
2. Inorder
3. Preorder
4. Postorder
Enter Your Option: 4
BST Traversal in POSTORDER:
40 50 30
Press a Key to Continue

1. Creation of BST

Binary Search Tree Operations

- 2. Inorder
- 3. Preorder
- 4. Postorder

Enter Your Option: