

BINARY SEARCH TREE: -

CODE: -

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct bstNode {
5      int data;
6      struct bstNode* left;
7      struct bstNode* right;
8  };
9
10 /* ----- Create a New Node ----- */
11 struct bstNode* createNode(int data) {
12     struct bstNode* newNode = (struct bstNode*) malloc(sizeof(struct bstNode));
13     if (newNode != NULL) {
14         newNode->data = data;
15         newNode->left = NULL;
16         newNode->right = NULL;
17     }
18     return newNode;
19 }
20
21 /* ----- Corrected Recursive Insert ----- */
22 struct bstNode* insertBSTRecursive(struct bstNode* root, int data) {
23
24     if (root == NULL) {
25         return createNode(data);
26     }
27
28     if (data < root->data) {
29         root->left = insertBSTRecursive(root->left, data);
30     }
31     else if (data > root->data) {
32         root->right = insertBSTRecursive(root->right, data);
33     }
34     else {
35         printf("\nDuplicate value entered - %d.\n", data);
36     }
37
38     return root;
39 }
40
41 /* ----- Iterative Insert ----- */
42 struct bstNode* insertBSTNode(struct bstNode* root, int data) {
43     struct bstNode* newNode = createNode(data);
44
45     if (root == NULL)
46         return newNode;
47
48     struct bstNode* current = root;
49
50     while (1) {
51         if (data < current->data) {
52             if (current->left == NULL) {
53                 current->left = newNode;
54                 break;
55             }
56             current = current->left;
57         }
58         else if (data > current->data) {
59             if (current->right == NULL) {
60                 current->right = newNode;
61                 break;
62             }
63             current = current->right;
64         }
65         else {
66             printf("%d already exists in the Binary Search Tree. Duplicate values are not allowed.\n", data);
67             free(newNode);
68             break;
69         }
70     }
71
72     return root;
73 }

```

```

75  /* ----- Tree Traversals ----- */
76  void displayInOrder(struct bstNode* root) {
77      if (root != NULL) {
78          displayInOrder(root->left);
79          printf(" %d ", root->data);
80          displayInOrder(root->right);
81      }
82  }
83
84  void displayPreOrder(struct bstNode* root) {
85      if (root != NULL) {
86          printf(" %d ", root->data);
87          displayPreOrder(root->left);
88          displayPreOrder(root->right);
89      }
90  }
91
92  void displayPostOrder(struct bstNode* root) {
93      if (root != NULL) {
94          displayPostOrder(root->left);
95          displayPostOrder(root->right);
96          printf(" %d ", root->data);
97      }
98  }
99
100 /* ----- User Menu ----- */
101 void binSearchTree() {
102
103     struct bstNode* root = NULL;
104     int choice = 1;
105
106     while (choice == 1) {
107         printf("\nEnter data for Binary Search Tree 0:Exit 1:Continue : ");
108         scanf("%d", &choice);

```

```

109
110     if (choice == 1) {
111         int data;
112         printf("Enter the data to be inserted : ");
113         scanf("%d", &data);
114
115         // Use any one of the insertion methods:
116         // root = insertBSTNode(root, data);
117         root = insertBSTRecursive(root, data);
118     }
119 }
120
121 printf("\nThe PreOrder traversal of the tree ... ");
122 displayPreOrder(root);
123
124 printf("\nThe InOrder traversal of the tree ... ");
125 displayInOrder(root);
126
127 printf("\nThe PostOrder traversal of the tree ... ");
128 displayPostOrder(root);
129 }
130
131 /* ----- Optional main() ----- */
132 /*
133 int main() {
134     binSearchTree();
135     return 0;
136 }
137 */
138 int main() {
139     binSearchTree();
140     return 0;
141 }
142

```

OUTPUT: -



C:\Users\student\1WA24CS18: X



Enter data for Binary Search Tree 0:Exit 1:Continue : 1
Enter the data to be inserted : 25

Enter data for Binary Search Tree 0:Exit 1:Continue : 1
Enter the data to be inserted : 36

Enter data for Binary Search Tree 0:Exit 1:Continue : 1
Enter the data to be inserted : 99

Enter data for Binary Search Tree 0:Exit 1:Continue : 0

The PreOrder traversal of the tree ... 25 36 99

The InOrder traversal of the tree ... 25 36 99

The PostOrder traversal of the tree ... 99 36 25

Process returned 0 (0x0) execution time : 10.814 s

Press any key to continue.

|