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

typedef struct tree {
    int data;
    struct tree *rlink, *llink;
} *TNODE;

TNODE getnode() {
    TNODE temp = (TNODE)malloc(sizeof(struct tree));
    if(temp == NULL) {
        printf("Out of memory!!!\n");
        return NULL;
    }
    return temp;
}

TNODE insert(TNODE root) {
    int n, ele, i, flag;
    TNODE temp, prev;

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

    for(i = 0; i < n; i++) {
        printf("Enter element to be inserted: ");
        scanf("%d", &ele);
        TNODE newN = getnode();
        newN->data = ele;
        newN->rlink = newN->llink = NULL;

        if(root == NULL) {
            root = newN;
            continue;
        }

        prev = NULL;
        temp = root;
        flag = 0;

        while(temp != NULL) {
            prev = temp;
            if(ele == temp->data) {
                printf("Redundant data\n");
                flag = 1;
                break;
            }
            if(ele < temp->data)
                temp = temp->llink;
            else
                temp = temp->rlink;
        }

```

```
54          if(flag == 1) continue;
55          if(ele < prev->data)
56              prev->llink = newN;
57          else
58              prev->rlink = newN;
59      }
60      return root;
61  }
62
63  void inorder(TNODE root) {
64      if(root != NULL) {
65          inorder(root->llink);
66          printf("%d\n", root->data);
67          inorder(root->rlink);
68      }
69  }
70
71  void preorder(TNODE root) {
72      if(root != NULL) {
73          printf("%d\n", root->data);
74          preorder(root->llink);
75          preorder(root->rlink);
76      }
77  }
78
79  void postorder(TNODE root) {
80      if(root != NULL) {
81          postorder(root->llink);
82          postorder(root->rlink);
83          printf("%d\n", root->data);
84      }
85  }
86
87  int search(TNODE root, int key) {
88      while(root != NULL) {
89          if(root->data == key)
90              return 1; // Successful search
91          if(key < root->data)
92              root = root->llink;
93          else
94              root = root->rlink;
95      }
96      return -1; // Unsuccessful search
97  }
98
99  void main() {
100     TNODE root = NULL;
101     int choice, ele, key, flag;
102
103     for(;;) {
104         printf("\nEnter:\n1. Insert\n2. Inorder\n3. Preorder\n4.
    Postorder\n5. Search\n6. Exit\n");
105         scanf("%d", &choice);
106
107         switch(choice) {
```

```
108             case 1:
109                 root = insert(root);
110                 break;
111             case 2:
112                 if(root == NULL) {
113                     printf("Tree is empty\n");
114                 } else {
115                     printf("The contents are:\n");
116                     inorder(root);
117                 }
118                 break;
119             case 3:
120                 if(root == NULL) {
121                     printf("Tree is empty\n");
122                 } else {
123                     printf("The contents are:\n");
124                     preorder(root);
125                 }
126                 break;
127             case 4:
128                 if(root == NULL) {
129                     printf("Tree is empty\n");
130                 } else {
131                     printf("The contents are:\n");
132                     postorder(root);
133                 }
134                 break;
135             case 5:
136                 printf("Enter the node to be searched:\n");
137                 scanf("%d", &key);
138                 flag = search(root, key);
139                 if(flag == -1)
140                     printf("Unsuccessful search!!!\n");
141                 else
142                     printf("Successful search!!!\n");
143                 break;
144             case 6:
145                 exit(0);
146             default:
147                 printf("Invalid choice. Try again.\n");
148         }
149     }
150 }
```