```c
// C program to find minimum and maximum value node in binar
// Tree.

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

/* A binary tree node has data, pointer to left child
and a pointer to right child */
struct node {
    int data;
    struct node* left;
    struct node* right;
};

/* Helper function that allocates a new node
with the given data and NULL left and right
pointers. */
struct node* newNode(int data)
{
    struct node* node
        = (struct node*)malloc(sizeof(struct node));
    node->data = data;
    node->left = NULL;
    node->right = NULL;

    return (node);
}

/* Give a binary search tree and a number,
inserts a new node with the given number in
the correct place in the tree. Returns the new
root pointer which the caller should then use
(the standard trick to avoid using reference
parameters). */
struct node* insert(struct node* node, int data)
{
    /* 1. If the tree is empty, return a new,
        single node */
    if (node == NULL)
```

```
40:         return (newNode(data));
41:     else {
42:         /* 2. Otherwise, recur down the tree */
43:         if (data <= node->data)
44:             node->left = insert(node->left, data);
45:         else
46:             node->right = insert(node->right, data);
47:
48:         /* return the (unchanged) node pointer */
49:         return node;
50:     }
51: }
52:
53: /* Given a non-empty binary search tree,
54: return the minimum data value found in that
55: tree. Note that the entire tree does not need
56: to be searched. */
57: int minValue(struct node* node)
58: {
59:     struct node* current = node;
60:
61:     /* loop down to find the leftmost leaf */
62:     while (current->left != NULL) {
63:         current = current->left;
64:     }
65:     return (current->data);
66: }
67:
68: int maxValue(struct node* node)
69: {
70:     struct node* current = node;
71:
72:     /* loop down to find the leftmost leaf */
73:     while (current->right != NULL) {
74:         current = current->right;
75:     }
76:     return (current->data);
77: }
78:
```

```
79: /* Driver code*/
80: int main()
81: {
82:     struct node* root = NULL;
83:     root = insert(root, 4);
84:     insert(root, 2);
85:     insert(root, 1);
86:     insert(root, 3);
87:     insert(root, 6);
88:     insert(root, 5);
89:
90:     // Function call
91:     printf("\n Minimum value in BST is %d", minValue(root));
92:     printf("\n Maximum value in BST is %d", maxValue(root));
93:     getchar();
94:     return 0;
95: }
96:
```