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

// Structure for a node of the binary search tree
struct Node {
    int data;
    struct Node* left;
    struct Node* right;
};

// Function to create a new node
struct Node* createNode(int value) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = value;
    newNode->left = newNode->right = NULL;
    return newNode;
}

// Function to insert a new node with given key in BST
struct Node* insert(struct Node* root, int value) {
    // If the tree is empty, return a new node
    if (root == NULL) return createNode(value);

    // Otherwise, recur down the tree
    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
        root->right = insert(root->right, value);

    // Return the (unchanged) node pointer
    return root;
```

```
}

// Function to perform inorder traversal of BST
void inorder(struct Node* root) {
   if (root != NULL) {
      inorder(root->left);
      printf("%d ", root->data);
      inorder(root->right);
   }
}

// Function to perform preorder traversal of BST
void preorder(struct Node* root) {
   if (root != NULL) {
      printf("%d ", root->data);
      preorder(root->left);
      preorder(root->right);
   }
}

// Function to perform postorder traversal of BST
void postorder(struct Node* root) {
   if (root != NULL) {
      postorder(root->left);
      postorder(root->right);
      printf("%d ", root->data);
   }
}

// Function to display elements in the tree
void display(struct Node* root) {
```

```
    printf("Inorder traversal: ");
    inorder(root);
    printf("\nPreorder traversal: ");
    preorder(root);
    printf("\nPostorder traversal: ");
    postorder(root);
}

int main() {
    struct Node* root = NULL;
    int elements[] = {50, 30, 70, 20, 40, 60, 80}; // Example elements

    // Construct the binary search tree
    for (int i = 0; i < sizeof(elements) / sizeof(elements[0]); i++) {
        root = insert(root, elements[i]);
    }

    // Display the elements in the tree using different traversal methods
    display(root);

    return 0;
}
```

```
Inorder traversal: 20 30 40 50 60 70 80
Preorder traversal: 50 30 20 40 70 60 80
Postorder traversal: 20 40 30 60 80 70 50
```