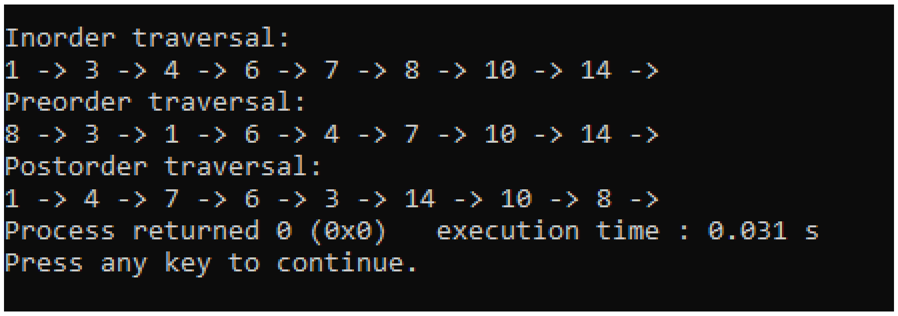
1. Write a program
    a) To construct a binary Search tree.
    b) To traverse the tree using all the methods i.e., in-order, preorder and post order
    c) To display the elements in the tree.

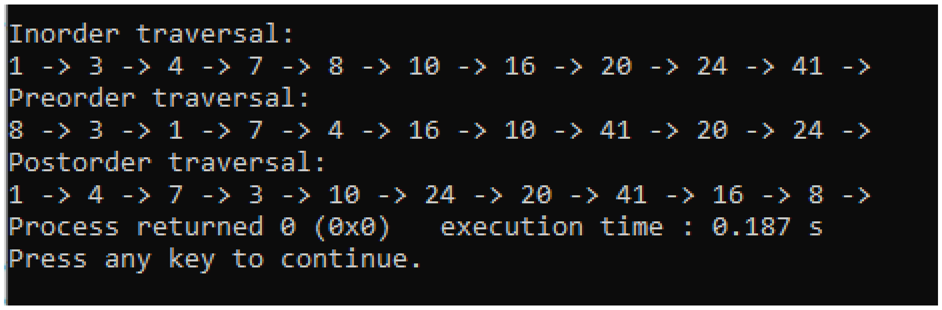
Code:

```
// Binary Search Tree operations in C

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

struct node {
  int key;
  struct node *left, *right;
};

// Create a node
struct node *newNode(int item) {
  struct node *temp = (struct node *)malloc(sizeof(struct node));
  temp->key = item;
  temp->left = temp->right = NULL;
  return temp;
}

// Inorder Traversal
void inorder(struct node *root) {
  if (root != NULL) {
    // Traverse left
    inorder(root->left);

    // Traverse root
    printf("%d -> ", root->key);

    // Traverse right
    inorder(root->right);
  }
}

// Preorder Traversal
void preorder(struct node *root) {
  if (root != NULL) {
    // Traverse root
    printf("%d -> ", root->key);
    // Traverse left
    preorder(root->left);
    // Traverse right
    preorder(root->right);
  }
}

// Postorder Traversal
void postorder(struct node *root) {
  if (root != NULL) {
    // Traverse left
    postorder(root->left);
    // Traverse right
    postorder(root->right);
    // Traverse root
    printf("%d -> ", root->key);
  }
}

// Insert a node
struct node *insert(struct node *node, int key) {
  // Return a new node if the tree is empty
  if (node == NULL) return newNode(key);

  // Traverse to the right place and insert the node
  if (key < node->key)
    node->left = insert(node->left, key);
  else
    node->right = insert(node->right, key);

  return node;
}

// Driver code
int main() {
  struct node *root = NULL;
  root = insert(root, 8);
  root = insert(root, 3);
  root = insert(root, 1);
  root = insert(root, 6);
  root = insert(root, 7);
  root = insert(root, 10);
  root = insert(root, 14);
  root = insert(root, 4);
  printf("\nInorder traversal: \n");
  inorder(root);

  printf("\nPreorder traversal: \n");
  preorder(root);

  printf("\nPostorder traversal: \n");
  postorder(root);

}
```

Output:



```
Inorder traversal:
1 -> 3 -> 4 -> 6 -> 7 -> 8 -> 10 -> 14 ->
Preorder traversal:
8 -> 3 -> 1 -> 6 -> 4 -> 7 -> 10 -> 14 ->
Postorder traversal:
1 -> 4 -> 7 -> 6 -> 3 -> 14 -> 10 -> 8 ->
Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```



```
Inorder traversal:
1 -> 3 -> 4 -> 7 -> 8 -> 10 -> 16 -> 20 -> 24 -> 41 ->
Preorder traversal:
8 -> 3 -> 1 -> 7 -> 4 -> 16 -> 10 -> 41 -> 20 -> 24 ->
Postorder traversal:
1 -> 4 -> 7 -> 3 -> 10 -> 24 -> 20 -> 41 -> 16 -> 8 ->
Process returned 0 (0x0)   execution time : 0.187 s
Press any key to continue.
```