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

struct node {
  int data;
  struct node *right_child;
  struct node *left_child;
};

 struct node* new_node(int x){
  struct node *temp;
  temp = malloc(sizeof(struct node));
  temp->data = x;
  temp->left_child = NULL;
  temp->right_child = NULL;

  return temp;
}

/*To create a new node*/
struct node* insert(struct node * root, int x){
  if (root == NULL)
    return new_node(x);
  else if (x > root->data)
    root->right_child = insert(root->right_child, x);
  else
    root -> left_child = insert(root->left_child, x);
  return root;
}

/*function to traverse the nodes of binary tree in preorder*/
void traversePreorder(struct node* root)
{
    if (root == NULL)
        return;
    printf(" %d ", root->data);
    traversePreorder(root->left_child);
    traversePreorder(root->right_child);
}


/*function to traverse the nodes of binary tree in Inorder*/
void traverseInorder(struct node* root)
{
    if (root == NULL)
        return;
    traverseInorder(root->left_child);
    printf(" %d ", root->data);
    traverseInorder(root->right_child);
}

/*function to traverse the nodes of binary tree in postorder*/
void traversePostorder(struct node* root)
{
    if (root == NULL)
        return;
    traversePostorder(root->left_child);
    traversePostorder(root->right_child);
    printf(" %d ", root->data);
}

int main() {
    struct node* root = NULL;  // Start with an empty tree
    int choice, value;

    while (1) {
        printf("\nBinary Search Tree Operations:\n");
        printf("1. create\n");
        printf("2. preorder\n");
        printf("3. Inorder Traversal\n");
        printf("4. postorder Traversal\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                root = insert(root,value);
                printf("Inserted %d into the BST.\n", value);
                break;
            case 2:
                traversePreorder(root);
                break;
            case 3:
                traverseInorder(root);
                break;
            case 4:
                traversePostorder(root);
                break;
            case 5:
                printf("Exiting...\n");
                exit(0);
            default:
                printf("Invalid choice! Please try again.\n");
        }
    }

    return 0;
}
```