BINARY SEARCH TREE:

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

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

// A utility function to create a new BST 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;
}

// A utility function to do inorder traversal of BST
void inorder(struct node* root)
{
    if (root != NULL) {
        inorder(root->left);
        printf("%d \t", root->key);
        inorder(root->right);
    }

}
```

```
void preorder(struct node* root)
{
    if (root != NULL) {
        printf("%d \t", root->key);
        preorder(root->left);
        preorder(root->right);
    }

}

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

}

struct node* insert(struct node* node, int key)
{

    if (node == NULL)
        return newNode(key);



    if (key < node->key)
        node->left = insert(node->left, key);
    else if (key > node->key)
        node->right = insert(node->right, key);
```

```
    return node;
}
struct node* minValueNode(struct node* node)
{
    struct node* current = node;

    /* loop down to find the leftmost leaf */
    while (current && current->left != NULL)
        current = current->left;

    return current;
}
struct node* deleteNode(struct node* root, int key)
{
    // base case
    if (root == NULL)
        return root;
    if (key < root->key)
        root->left = deleteNode(root->left, key);
    else if (key > root->key)
        root->right = deleteNode(root->right, key);
 else {
        // node with only one child or no child
        if (root->left == NULL) {
            struct node* temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL) {
```

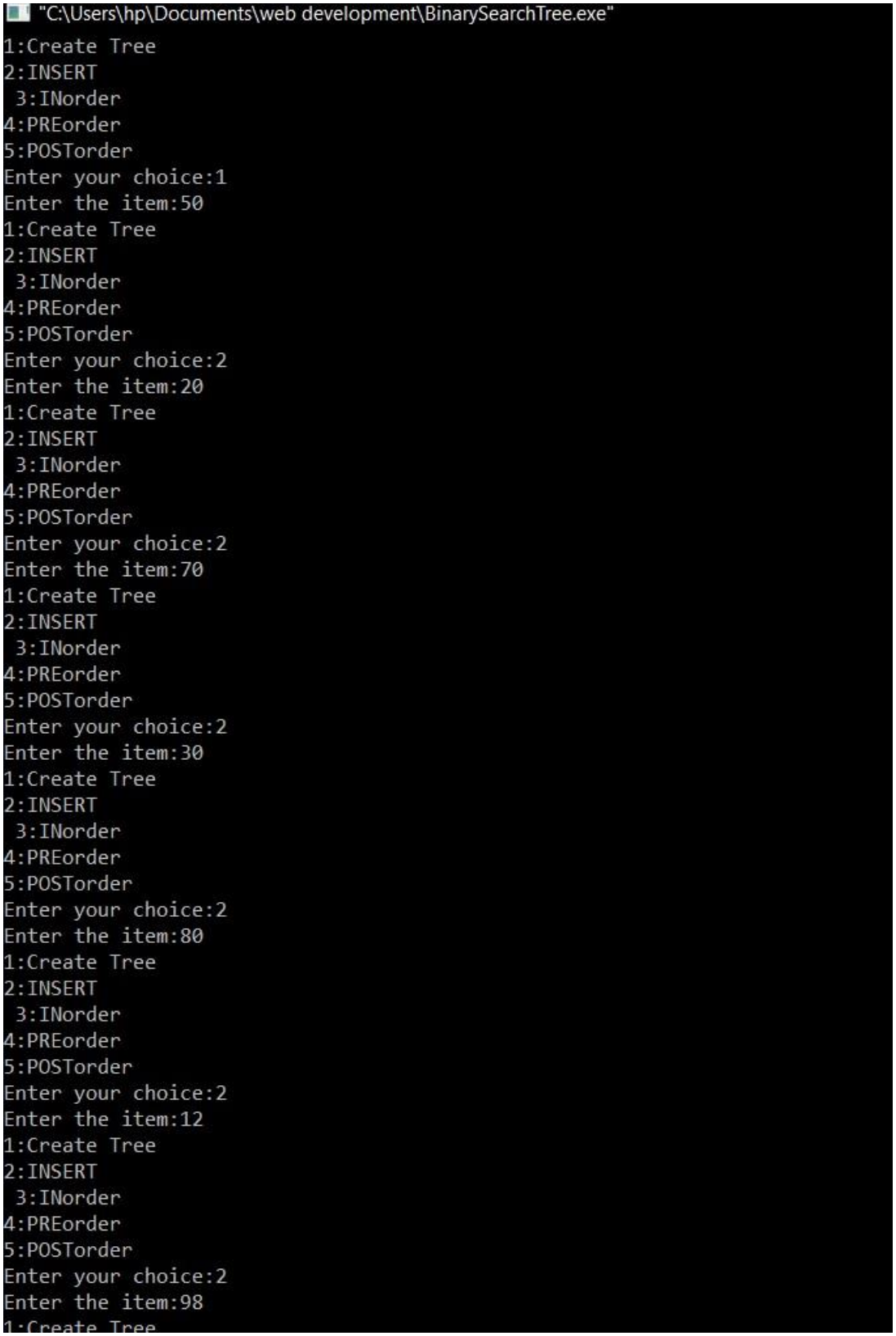
```
            struct node* temp = root->left;
            free(root);
            return temp;
        }
        struct node* temp = minValueNode(root->right);
  root->key = temp->key;
  root->right = deleteNode(root->right, temp->key);
    }
    return root;
}

int main()
{  int ch;
   int item; struct node* root = NULL;
   for(;;)
   {
       printf("1:Create Tree \n2:INSERT \n 3:INorder \n4:PREorder
\n5:POSTorder  \n");
       printf("Enter your choice:");
       scanf("%d",&ch);
       switch(ch)
       {
          case 1:   printf("Enter the item:");
                    scanf("%d",&item);
                     root=insert(root, item);
                     break;
           case 2:  printf("Enter the item:");
                    scanf("%d",&item);
                     insert(root, item);
                     break;
          case 3: inorder(root);
```

```
                break;
        case 4: preorder(root);
                break;
        case 5: postorder(root);
                 break;
        case 6:printf("Enter the item:");
                scanf("%d",&item);
               deleteNode(root,item);
               break;
        default: break;
     }
   }

    return 0;
}
```

OUTPUT:


"C:\Users\hp\Documents\web development\BinarySearchTree.exe"
1:Create Tree
2:INSERT
3:INorder
4:PREorder
5:POSTorder
Enter your choice:1
Enter the item:50
1:Create Tree
2:INSERT
3:INorder
4:PREorder
5:POSTorder
Enter your choice:2
Enter the item:20
1:Create Tree
2:INSERT
3:INorder
4:PREorder
5:POSTorder
Enter your choice:2
Enter the item:70
1:Create Tree
2:INSERT
3:INorder
4:PREorder
5:POSTorder
Enter your choice:2
Enter the item:30
1:Create Tree
2:INSERT
3:INorder
4:PREorder
5:POSTorder
Enter your choice:2
Enter the item:80
1:Create Tree
2:INSERT
3:INorder
4:PREorder
5:POSTorder
Enter your choice:2
Enter the item:12
1:Create Tree
2:INSERT
3:INorder
4:PREorder
5:POSTorder
Enter your choice:2
Enter the item:98
1:Create Tree

```
 3:INorder
4:PREorder
5:POSTorder
Enter your choice:2
Enter the item:12
1:Create Tree
2:INSERT
 3:INorder
4:PREorder
5:POSTorder
Enter your choice:2
Enter the item:98
1:Create Tree
2:INSERT
 3:INorder
4:PREorder
5:POSTorder
Enter your choice:3
12      20      30      50      70      80      98      1:Create Tree
2:INSERT
 3:INorder
4:PREorder
5:POSTorder
Enter your choice:4
50      20      12      30      70      80      98      1:Create Tree
2:INSERT
 3:INorder
4:PREorder
5:POSTorder
Enter your choice:5
12      30      20      98      80      70      50      1:Create Tree
2:INSERT
 3:INorder
4:PREorder
5:POSTorder
Enter your choice:
```