Binary tree

Binary Tree:

Code:

```
#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

    int data;

    struct Node *left;

    struct Node *right;

} Node;

Node* createNode(int data) {

    Node* newNode = (Node*)malloc(sizeof(Node));

    newNode->data = data;

    newNode->left = NULL;

    newNode->right = NULL;

    return newNode;

}

Node* insertNode(Node* root, int data) {

    if (root == NULL) {

        return createNode(data);

    }

    if (data < root->data) {

        root->left = insertNode(root->left, data);

    } else {

        root->right = insertNode(root->right, data);

    }

    return root;

}

void inorderTraversal(Node* root) {

    if (root != NULL) {

        inorderTraversal(root->left);

        printf("%d ", root->data);

        inorderTraversal(root->right);

    }
```

```
}
void preorderTraversal(Node* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorderTraversal(root->left);
        preorderTraversal(root->right);
    }
}
void postorderTraversal(Node* root) {
    if (root != NULL) {
        postorderTraversal(root->left);
        postorderTraversal(root->right);
        printf("%d ", root->data);
    }
}
void freeTree(Node* root) {
    if (root != NULL) {
        freeTree(root->left);
        freeTree(root->right);
        free(root);
    }
}
int main() {
    Node* root = NULL;
    root = insertNode(root, 50);
    insertNode(root, 30);
    insertNode(root, 20);
    insertNode(root, 40);
    insertNode(root, 70);
    insertNode(root, 60);
    insertNode(root, 80);
    printf("In-order traversal of the binary tree:\n");
    inorderTraversal(root);
    printf("\n");
    printf("Pre-order traversal of the binary tree:\n");
```

```
preorderTraversal(root);

printf("\n");

printf("Post-order traversal of the binary tree:\n");

postorderTraversal(root);

printf("\n");

freeTree(root);

return 0;

}
```

Output:

In-order traversal of the binary tree:

20 30 40 50 60 70 80

Pre-order traversal of the binary tree:

50 30 20 40 70 60 80

Post-order traversal of the binary tree:

20 40 30 60 80 70 50

Binary Search tree:

Code:

```
#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

    int data;

    struct Node *left;

    struct Node *right;

} Node;

Node* createNode(int data) {

    Node* newNode = (Node*)malloc(sizeof(Node));

    newNode->data = data;

    newNode->left = NULL;

    newNode->right = NULL;

    return newNode;

}

Node* insertNode(Node* root, int data) {

    if (root == NULL) {

        return createNode(data);
```

```
}


if (data < root->data) {
    root->left = insertNode(root->left, data);
} else if (data > root->data) {
    root->right = insertNode(root->right, data);
}
    return root;
}
Node* searchNode(Node* root, int data) {
    if (root == NULL || root->data == data) {
        return root;
    }

    if (data < root->data) {
        return searchNode(root->left, data);
    } else {
        return searchNode(root->right, data);
    }
}
Node* findMin(Node* root) {
    while (root->left != NULL) {
        root = root->left;
    }
    return root;
}
Node* deleteNode(Node* root, int data) {
if (root == NULL) {
    return root;
}
if (data < root->data) {
    root->left = deleteNode(root->left, data);
} else if (data > root->data) {
    root->right = deleteNode(root->right, data);
```

```
} else {

    if (root->left == NULL) {
        Node* temp = root->right;
        free(root);
        return temp;
    } else if (root->right == NULL) {
        Node* temp = root->left;
        free(root);
        return temp;
      }
        Node* temp = findMin(root->right);
        root->data = temp->data;
        root->right = deleteNode(root->right, temp->data);
    }


    return root;
}
void inorderTraversal(Node* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
    }
}
int main() {
    Node* root = NULL;
    root = insertNode(root, 50);
    insertNode(root, 30);
    insertNode(root, 20);
    insertNode(root, 40);
    insertNode(root, 70);
    insertNode(root, 60);
    insertNode(root, 80);
    printf("In-order traversal of the binary search tree:\n");
```

```
inorderTraversal(root);

printf("\n");

int searchValue = 40;

Node* searchResult = searchNode(root, searchValue);

if (searchResult != NULL) {

    printf("Node with value %d found in the tree.\n", searchValue);

} else {

        printf("Node with value %d not found in the tree.\n", searchValue);

    }

    int deleteValue = 30;

    root = deleteNode(root, deleteValue);

    printf("In-order traversal after deleting node %d:\n", deleteValue);

    inorderTraversal(root);

    printf("\n");

    return 0;

}
```

Output:

In-order traversal of the binary search tree:

20 30 40 50 60 70 80

In-order traversal after deleting node 30:

20 40 50 60 70 80

Binary tree traversal:

Code:

```
#include <stdio.h>

#include <stdlib.h>

typedef struct Node {

    int data;

    struct Node *left;

    struct Node *right;

} Node;

Node* createNode(int data) {

    Node* newNode = (Node*)malloc(sizeof(Node));
```

```
    newNode->data = data;
    newNode->left = NULL;
    newNode->right = NULL;
    return newNode;
}
Node* insertNode(Node* root, int data) {
    if (root == NULL) {
        return createNode(data);
    }

    if (data < root->data) {
        root->left = insertNode(root->left, data);
    } else {
        root->right = insertNode(root->right, data);
    }

    return root;
}
void inorderTraversal(Node* root) {
    if (root != NULL) {
        inorderTraversal(root->left);
        printf("%d ", root->data);
        inorderTraversal(root->right);
    }
}
void preorderTraversal(Node* root) {
    if (root != NULL) {
        printf("%d ", root->data);
        preorderTraversal(root->left);
        preorderTraversal(root->right);
    }
}
void postorderTraversal(Node* root) {
    if (root != NULL) {
        postorderTraversal(root->left);
```

```
        postorderTraversal(root->right);
        printf("%d ", root->data);
    }
}
int main() {
    Node* root = NULL;
    root = insertNode(root, 50);
    insertNode(root, 30);
    insertNode(root, 20);
    insertNode(root, 40);
    insertNode(root, 70);
    insertNode(root, 60);
    insertNode(root, 80);
    printf("In-order traversal of the binary tree:\n");
    inorderTraversal(root);
    printf("\n");
    printf("Pre-order traversal of the binary tree:\n");
    preorderTraversal(root);
    printf("\n");
    printf("Post-order traversal of the binary tree:\n");
    postorderTraversal(root);
    printf("\n");
    return 0;
}
```

Output:

In-order traversal of the binary tree:

20 30 40 50 60 70 80

Pre-order traversal of the binary tree:

50 30 20 40 70 60 80

Post-order traversal of the binary tree:

20 40 30 60 80 70 50