# Assignment 10: Binary Search Tree

Name: Aarya Gawade
UEC No.: UEC2023122
Batch: A2

Code:

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

typedef struct bst {
    int data;
    struct bst* left;
    struct bst* right;
} node;

node* getnode() {
    node* temp = (node*)malloc(sizeof(node));
    if (temp == NULL) {
        printf("Memory allocation failed!\n");
        exit(1);
    }
    temp->left = NULL;
    temp->right = NULL;

    printf("Enter new data: ");
    scanf("%d", &temp->data);
    return temp;
}

node* createbst(node* root) {
    node* newNode = NULL;
    node* oldNode = NULL;
    char ch;
```

```
    do {
        newNode = getnode();
        if (root == NULL) {
            root = newNode;
        } else {
            oldNode = root;
            while (1) {
                if (newNode->data < oldNode->data) {
                    if (oldNode->left == NULL) {
                        oldNode->left = newNode;
                        break;
                    } else {
                        oldNode = oldNode->left;
                    }
                } else if (newNode->data > oldNode->data) {
                    if (oldNode->right == NULL) {
                        oldNode->right = newNode;
                        break;
                    } else {
                        oldNode = oldNode->right;
                    }
                } else {
                    printf("Duplicate node can't be created.\n");
                    free(newNode);
                    break;
                }
            }
        }

        printf("Do you want to create more nodes? (y/n): ");
        scanf(" %c", &ch); // Note the space before %c to consume leftover
newline.
    } while (ch != 'n');
    return root;
}

void inorder(node* node) {
    if (node == NULL) return;
    inorder(node->left);
    printf("%d ", node->data);
```

```
    inorder(node->right);
}

void preorder(node* node) {
    if (node == NULL) return;
    printf("%d ", node->data);
    preorder(node->left);
    preorder(node->right);
}

void postorder(node* node) {
    if (node == NULL) return;
    postorder(node->left);
    postorder(node->right);
    printf("%d ", node->data);
}

node* search(node* p, int key) {
    while (p != NULL) {
        if (key == p->data)
            return p;
        else if (key > p->data)
            p = p->right;
        else
            p = p->left;
    }
    return NULL;
}

int main() {
    node* root = NULL;
    int ch = 0, key = 0;

    do {
        printf("\n1. Create tree\n");
        printf("2. Inorder with recursion\n");
        printf("3. Preorder with recursion\n");
        printf("4. Postorder with recursion\n");
        printf("5. Search\n");
        printf("6. Exit\n");
```

```
        printf("Enter choice: ");
        scanf("%d", &ch);

        switch (ch) {
            case 1:
                root = createbst(root);
                break;

            case 2:
                printf("Inorder traversal: ");
                inorder(root);
                printf("\n");
                break;

            case 3:
                printf("Preorder traversal: ");
                preorder(root);
                printf("\n");
                break;

            case 4:
                printf("Postorder traversal: ");
                postorder(root);
                printf("\n");
                break;

            case 5:
                printf("Enter key to be searched: ");
                scanf("%d", &key);
                if (search(root, key))
                    printf("%d Found\n", key);
                else
                    printf("%d Not found\n", key);
                break;

            case 6:
                printf("Exiting program.\n");
                break;

            default:
```

```
                printf("Invalid choice! Please try again.\n");
        }
    } while (ch != 6);

    return 0;
}
```

## Output:

D:\OneDrive\Dokumen\Clg_work>cd "d:\OneDrive\Dokumen\Clg_work\Assignments\" && gcc 10bst.c -o 10bst && "d:\OneDrive\Dokumen\Clg_work\Assignments\"10bst

1. Create tree
2. Inorder with recursion
3. Preorder with recursion
4. Postorder with recursion
5. Search
6. Exit
Enter choice: 1
Enter new data: 1
Do you want to create more nodes? (y/n): y
Enter new data: 2
Do you want to create more nodes? (y/n): y
Enter new data: 3
Do you want to create more nodes? (y/n): y
Enter new data: 4
Do you want to create more nodes? (y/n): y
Enter new data: 5
Do you want to create more nodes? (y/n): y
Enter new data: 6
Do you want to create more nodes? (y/n): y
Enter new data: 7
Do you want to create more nodes? (y/n): n

1. Create tree
2. Inorder with recursion
3. Preorder with recursion
4. Postorder with recursion
5. Search

6. Exit
Enter choice: 2
Inorder traversal: 1 2 3 4 5 6 7

1. Create tree
2. Inorder with recursion
3. Preorder with recursion
4. Postorder with recursion
5. Search
6. Exit
Enter choice: 3
Preorder traversal: 1 2 3 4 5 6 7

1. Create tree
2. Inorder with recursion
3. Preorder with recursion
4. Postorder with recursion
5. Search
6. Exit
Enter choice: 4
Postorder traversal: 7 6 5 4 3 2 1

1. Create tree
2. Inorder with recursion
3. Preorder with recursion
4. Postorder with recursion
5. Search
6. Exit
Enter choice: 5
Enter key to be searched: 7
7 Found

1. Create tree
2. Inorder with recursion
3. Preorder with recursion
4. Postorder with recursion
5. Search
6. Exit
Enter choice: 6
Exiting program.