21BDS0340

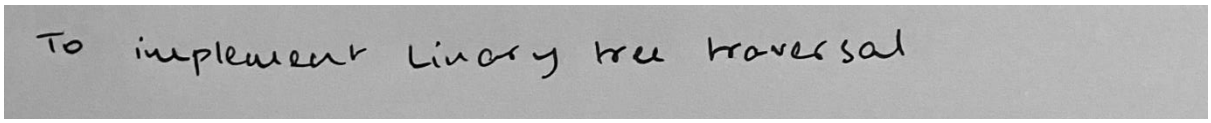
Abhinav Dinesh Srivatsa

Data Structures and Algorithms Lab

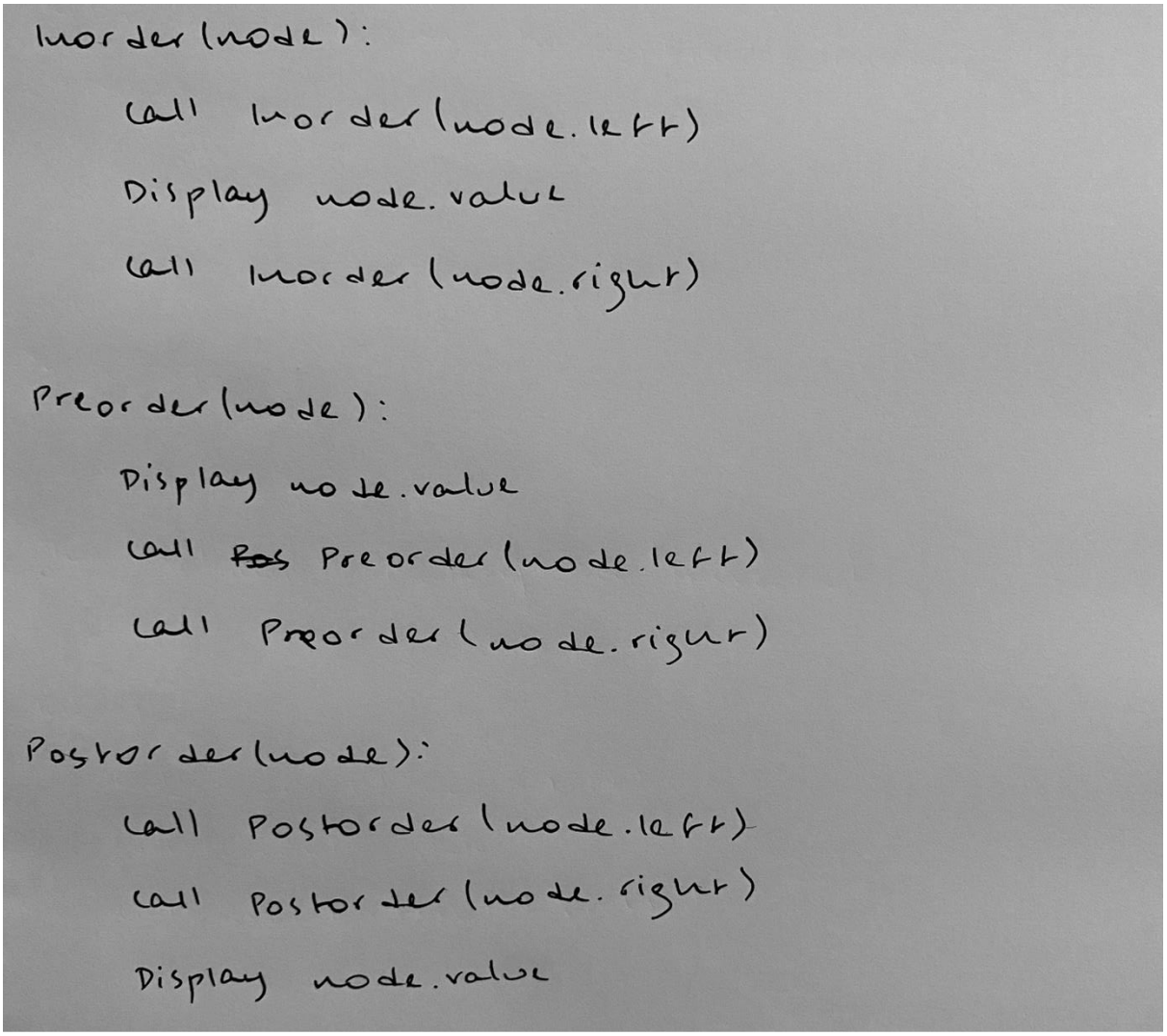
Assignment – III

1. Binary Tree Traversal

Aim



Algorithm



Code

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

```c
struct Node
{
    int val;
    struct Node *right;
    struct Node *left;
};

struct Node *addNode(int val)
{
    struct Node *root = malloc(sizeof(struct Node));
    root->val = val;
    root->left = NULL;
    root->right = NULL;
    return root;
}

struct Node *insert(int *arr, int i, int n)
{
    struct Node *root = NULL;
    if (i < n)
    {
        root = addNode(*(arr + i));
        root->left = insert(arr, 2 * i + 1, n);
        root->right = insert(arr, 2 * i + 2, n);
    }
    return root;
}

void preorder(struct Node *root)
{
    if (root == NULL)
        return;
    printf("%d ", root->val);
    preorder(root->left);
    preorder(root->right);
}

void postorder(struct Node *root)
{
    if (root == NULL)
        return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->val);
}

void inorder(struct Node *root)
```

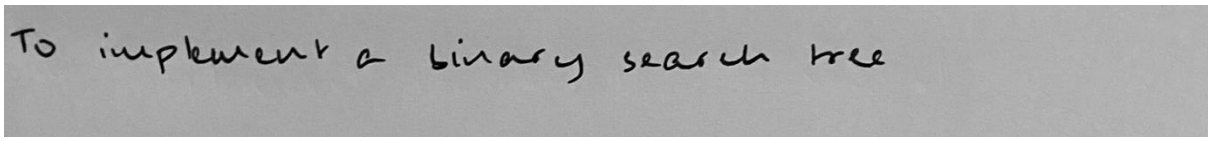
```
{
    if (root == NULL)
        return;
    inorder(root->left);
    printf("%d ", root->val);
    inorder(root->right);
}

int main()
{
    int n;
    printf("Enter number of values: ");
    scanf("%d", &n);
    int *arr = malloc(sizeof(int) * n);
    for (int x = 0; x < n; x++)
        scanf("%d", arr + x);
    struct Node *root = insert(arr, 0, n); // creates complete binary tree
    inorder(root);
    printf("\n");
    preorder(root);
    printf("\n");
    postorder(root);
    free(arr);
}
```

Output

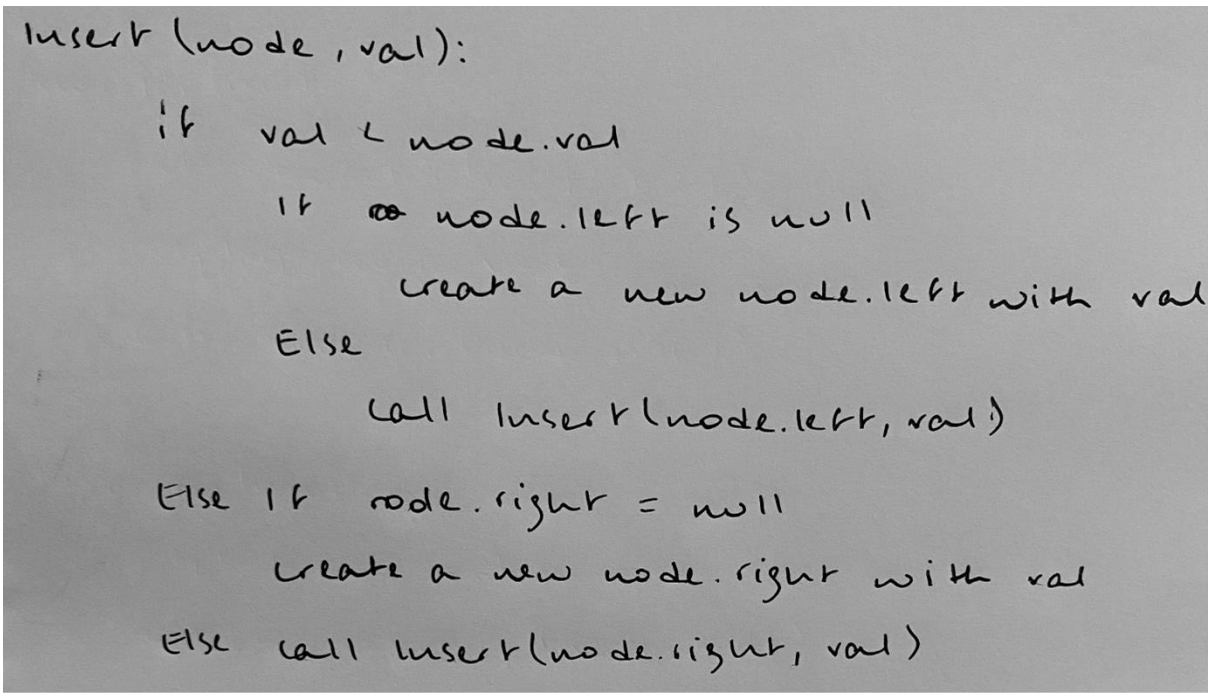
```
Enter number of values: 10
1
2
3
4
5
6
7
8
9
10
8 4 9 2 10 5 1 6 3 7
1 2 4 8 9 5 10 3 6 7
8 9 4 10 5 2 6 7 3 1
```

2. Binary Search Tree

Aim



Algorithm



Code

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

struct Node
{
    int val;
    struct Node *right;
    struct Node *left;
};

void preorder(struct Node *root)
{
    if (root == NULL)
        return;
    printf("%d ", root->val);
    preorder(root->left);
```

```
    preorder(root->right);
}

void postorder(struct Node *root)
{
    if (root == NULL)
        return;
    postorder(root->left);
    postorder(root->right);
    printf("%d ", root->val);
}

void inorder(struct Node *root)
{
    if (root == NULL)
        return;
    inorder(root->left);
    printf("%d ", root->val);
    inorder(root->right);
}

struct Node *addNode(int val)
{
    struct Node *root = malloc(sizeof(struct Node));
    root->val = val;
    root->left = NULL;
    root->right = NULL;
    return root;
}

void insert(struct Node *root, int val)
{
    if (val < root->val)
        if (root->left == NULL)
            root->left = addNode(val);
        else
            insert(root->left, val);
    else if (root->right == NULL)
        root->right = addNode(val);
    else
        insert(root->right, val);
}

int main()
{
    int n;
    printf("Enter number of values: ");
    scanf("%d", &n);
```

```
    int *arr = malloc(sizeof(int) * n);
    for (int x = 0; x < n; x++)
        scanf("%d", arr + x);
    struct Node *root = addNode(*arr);
    for (int x = 1; x < n; x++)
        insert(root, *(arr + x));
    inorder(root);
}
```

Output

```
Enter number of values: 10
1
2
3
4
5
6
7
8
9
10
1 2 3 4 5 6 7 8 9 10
Enter number of values: 10
6
3
9
2
8
0
9
2
7
3
0 2 2 3 3 6 7 8 9 9
```

3. Graph Traversal – DFS and BFS Algorithm

Aim

To implement depth and breadth first search algorithms for a graph

Algorithm

Create stack for DFS and queue for BFS

Insert means push/enqueue

Delete means pop / dequeue

Peek means peek/peek

Each node in a graph has its value and a list of next nodes

→ start at root of graph (any node)

check if root has already been explored

If not, go to it's next elements and insert them into the list (stack/queue)

start again with the peek of the list

Explanation:

Lists that are made with a stack operate on FILO, that means the latest node will be explored, hence depth first search

Queues operate on FIFO, this means that the oldest inserted node will be explored, hence breadth first search