Day 11

Program 56 ::- AVL tree implementation

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

typedef struct Treenode {
    int data;
    struct Treenode *left, *right;
} Treenode;

typedef struct {
    Treenode *root;
} Tree;

Treenode* newTreenode(int data) {
    Treenode* node = (Treenode*)malloc(sizeof(Treenode));
    node->data = data;
    node->left = node->right = NULL;
    return node;
}

int height(Treenode *root) {
    if (root == NULL)
        return 0;
    int left_height = height(root->left);
    int right_height = height(root->right);
    return (left_height > right_height ? left_height : right_height) + 1;
}

int getcol(int h) {
    if (h == 1)
        return 1;
    return getcol(h - 1) + getcol(h - 1) + 1;
}

void printTree(int **M, Treenode *root, int col, int row, int height) {
    if (root == NULL)
        return;
    M[row][col] = root->data;
    printTree(M, root->left, col - pow(2, height - 2), row + 1, height - 1);
    printTree(M, root->right, col + pow(2, height - 2), row + 1, height - 1);
}

void TreePrinter(Tree tree) {
    int h = height(tree.root);
```

```
    int col = getcol(h);
    int **M = (int **)malloc(h * sizeof(int *));
    for (int i = 0; i < h; i++) {
        M[i] = (int *)malloc(col * sizeof(int));
        for (int j = 0; j < col; j++) {
            M[i][j] = 0;
        }
    }
    printTree(M, tree.root, col / 2, 0, h);
    for (int i = 0; i < h; i++) {
        for (int j = 0; j < col; j++) {
            if (M[i][j] == 0)
                printf("  ");
            else
                printf("%d ", M[i][j]);
        }
        printf("\n");
    }
    for (int i = 0; i < h; i++) {
        free(M[i]);
    }
    free(M);
}

Treenode* insertLevelOrder(int arr[], Treenode* root, int i, int n) {
    if (i < n) {
        Treenode *temp = newTreenode(arr[i]);
        root = temp;
        root->left = insertLevelOrder(arr, root->left, 2 * i + 1, n);
        root->right = insertLevelOrder(arr, root->right, 2 * i + 2, n);
    }
    return root;
}

int main() {
    Tree myTree;
    myTree.root = NULL;

    int n;
    printf("Enter the number of nodes in the tree: ");
    scanf("%d", &n);

    int *arr = (int *)malloc(n * sizeof(int));
    printf("Enter the nodes in level order:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }
```

```
    myTree.root = insertLevelOrder(arr, myTree.root, 0, n);

    printf("Tree structure:\n");
    TreePrinter(myTree);

    free(arr);
    return 0;
}
```

```
Output :
Enter the number of nodes in the tree: 6
Enter the nodes in level order:
1
5
6
7
3
9
Tree structure:
      1
  5       6
7   3   9
```