**Assignment-1**

**Name of Student: Ayush Sanjay Dhangar**

**Batch: 02 Class: SY\_IT-A Roll No: 42**

**PRN: 12210406**

**Problem Statement:**

Write C or C++ program to represent binary tree or BST using array and linked list

**Binary tree using array:**

**Code:**

#include <stdio.h>

char tree[100];

int root(char key) {

    if (tree[0] != '\0')

        printf("Tree already has root\n");

    else

        tree[0] = key;

    return 0;

}

int left\_set(char key, int parent) {

    if (tree[parent] == '\0')

        printf("Can't set child at %d, no parent found\n", (parent \* 2) + 1);

    else

        tree[(parent \* 2) + 1] = key;

    return 0;

}

int right\_set(char key, int parent) {

    if (tree[parent] == '\0')

        printf("Can't set child at %d, no parent found\n", (parent \* 2) + 2);

    else

        tree[(parent \* 2) + 2] = key;

    return 0;

}

int print\_tree() {

    printf("\n");

    for (int i = 0; i < 30; i++) {

        if (tree[i] != '\0')

            printf("%c", tree[i]);

        else

            printf("-");

    }

    printf("\n");

    return 0;

}

int main() {

    int choice, parent;

    char key;

    do {

        printf("\nBinary Tree using array:\n");

        printf("1. Set Root\n");

        printf("2. Set Left Child\n");

        printf("3. Set Right Child\n");

        printf("4. Print Tree\n");

        printf("5. Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice) {

            case 1:

                printf("Enter the root element: ");

                scanf(" %c", &key);

                root(key);

                printf("\nRoot element inserted\n");

                break;

            case 2:

                printf("Enter the parent index: ");

                scanf("%d", &parent);

                printf("Enter the element to be inserted at left of index %d: ",parent);

                scanf(" %c", &key);

                left\_set(key, parent);

                printf("\n%c inserted to the left of index %d\n", key, parent);

                break;

            case 3:

                printf("Enter the parent index: ");

                scanf("%d", &parent);

                printf("Enter the element to be inserted at right of index %d: ",parent);

                scanf(" %c", &key);

                right\_set(key, parent);

                printf("\n%c inserted to the right of index %d\n", key, parent);

                break;

            case 4:

                printf("Printing tree: ");

                print\_tree();

                break;

            case 5:

                printf("Exiting program...\n");

                break;

            default:

                printf("Invalid choice, please try again\n");

        }

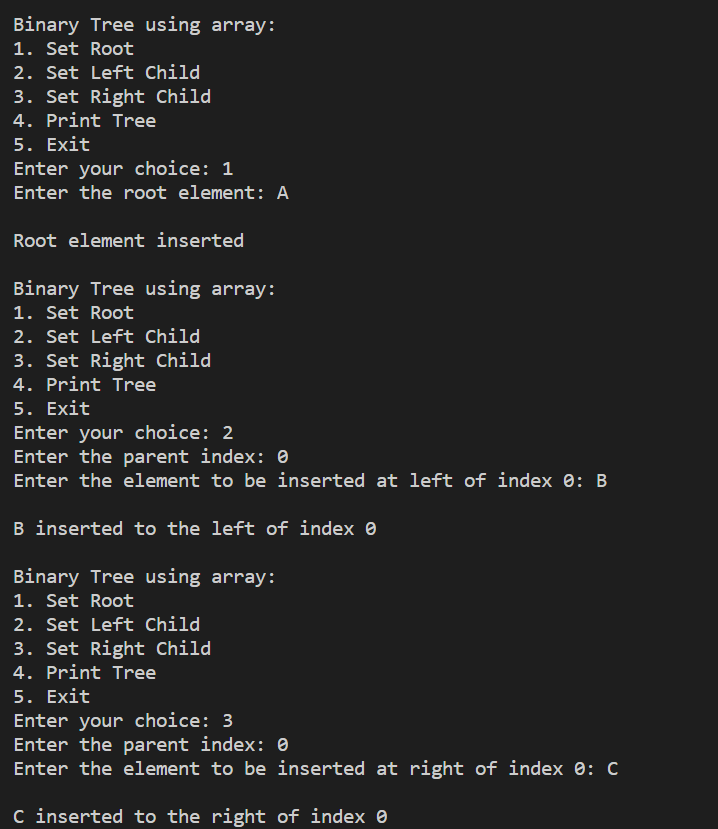
    } while (choice != 5);

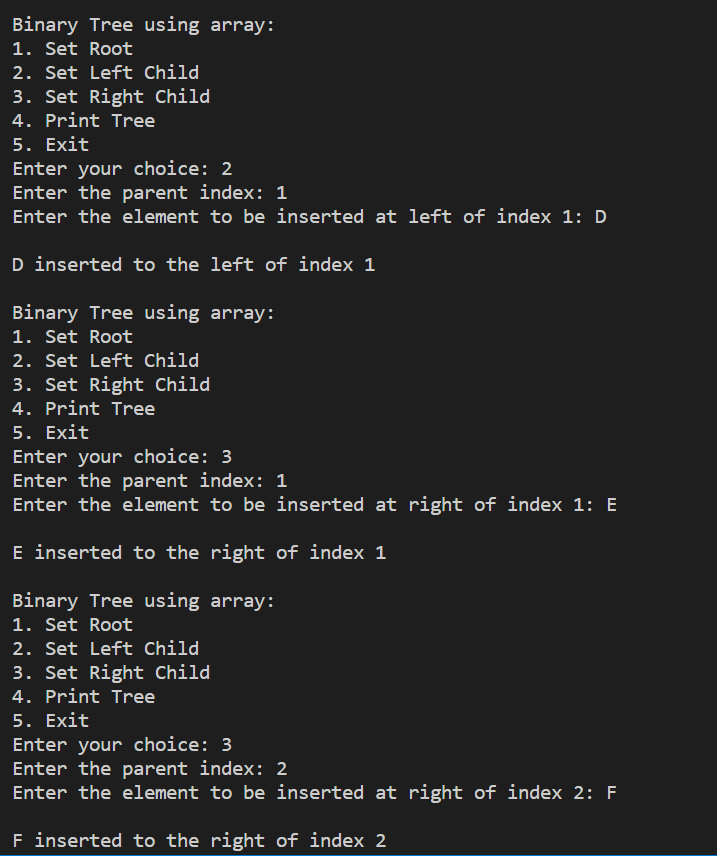
    return 0;

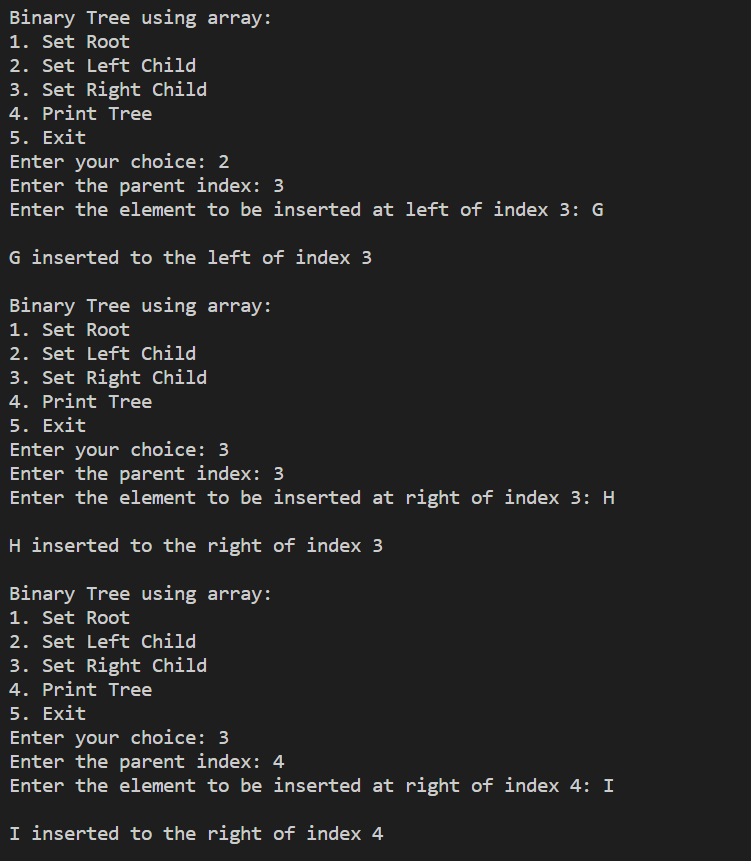
}

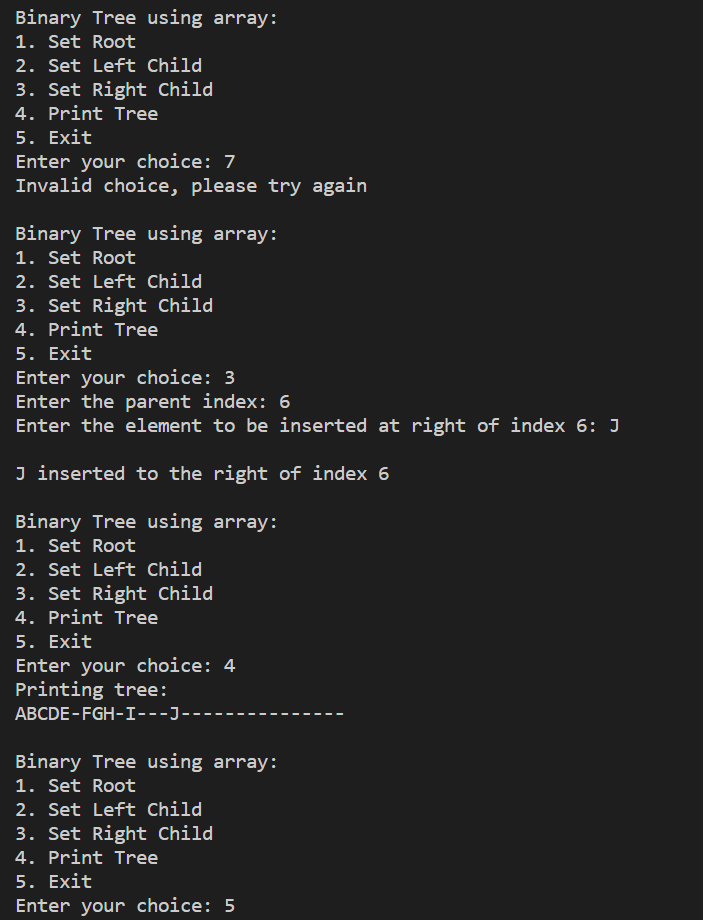
**Result:**

**Actual Output:**

****

****

****

****

**Binary tree using linked list:**

**Code:**

#include <stdio.h>

#include <stdlib.h>

struct binarytree

{

    struct binarytree \*left;

    int data;

    struct binarytree \*right;

};

struct binarytree \*root = NULL;

struct binarytree \*createTree()

{

    struct binarytree \*newnode;

    newnode = (struct binarytree \*)malloc(sizeof(struct binarytree));

    int data;

    printf("Enter data to be inserted in the binary tree: ");

    scanf("%d", &data);

    newnode->data = data;

    char choice;

    printf("\nDo you want to add a left node to %d? Y/N: ", newnode->data);

    scanf(" %c", &choice);

    if (choice == 'Y')

    {

        printf("Enter left child of the %d:\n", newnode->data);

        newnode->left = createTree();

    }

    else

    {

        newnode->left = NULL;

    }

    printf("\nDo you want to add a right node to %d? Y/N: ", newnode->data);

    scanf(" %c", &choice);

    if (choice == 'Y')

    {

        printf("Enter right child of the %d:\n", newnode->data);

        newnode->right = createTree();

    }

    else

    {

        newnode->right = NULL;

    }

    return newnode;

}

void inOrderTraversal(struct binarytree \*root)

{

    if (root != NULL)

    {

        inOrderTraversal(root->left);

        printf("%d ", root->data);

        inOrderTraversal(root->right);

    }

}

int main()

{

    printf("Binary tree using linked list\n");

    root = createTree();

    printf("Inorder traversal of the tree: ");

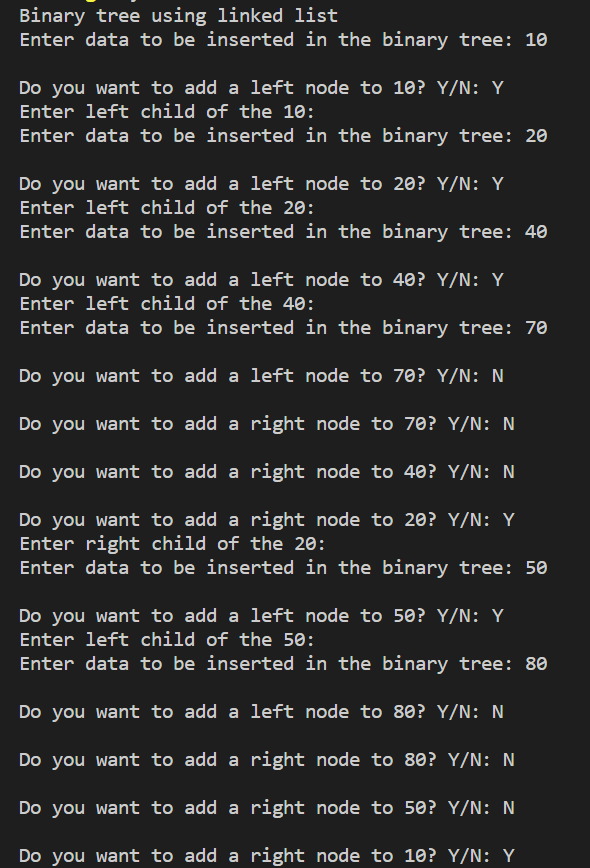
    inOrderTraversal(root);

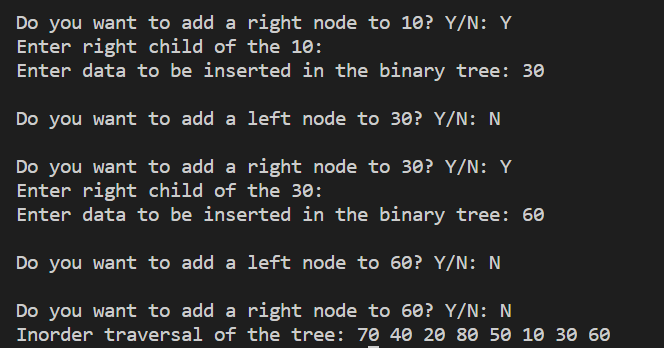
    return 0;

}

**Result:**

**Actual Output:**

****

****