**Name – Akshat Jaiswal**

**Roll No. – 21052646**

**Section – CSE 37**

**DSA LAB 6**

1. **WAP to implement Binary Tree using array and display all the nodes using another function.**

**Input:**

// PROGRAMED BY AKSHAT JAISWAL

#include <stdio.h>

*void* buildtree(*int* *arr*[], *int* *index*, *int* *value*)

{

*int* data;

*char* ch;

*arr*[*index*] = *value*;

    printf("\nDo you want to add Left child of %d?\n(Y/N): ", *value*);

    scanf("%c", &ch);

    fflush(stdin);

    if (ch == 'Y' || ch == 'y')

    {

        printf("\nEnter left child: ");

        scanf("%d", &data);

        fflush(stdin);

        buildtree(*arr*, 2 \* *index* + 1, data);

    }

    printf("\nDo you want to add Right child of %d?\n(Y/N): ", *value*);

    scanf("%c", &ch);

    fflush(stdin);

    if (ch == 'Y' || ch == 'y')

    {

        printf("\nEnter the Right child: ");

        scanf("%d", &data);

        fflush(stdin);

        buildtree(*arr*, 2 \* *index* + 2, data);

    }

}

*void* traversal(*int* *t*[])

{

    for (*int* i = 0; i < 8; i++)

    {

        if (*t*[i] == -1)

            printf("- ");

        else

        {

            printf("%d ", *t*[i]);

        }

    }

}

*void* Node(*int* *t*[])

{

    printf("ROOT ");

*int* i = 1;

*int* j = 2;

    while (i < 8 && j < 8)

    {

        if (i % 2 != 0)

        {

            printf("LC ");

        }

        i = 2 \* i + 1;

        if (j % 2 == 0)

        {

            printf("RC ");

        }

        j = 2 \* j + 2;

    }

}

*int* main(*void*)

{

*int* arr[8];

*int* value;

    for (*int* i = 0; i < 8; i++)

    {

        arr[i] = -1;

    }

    printf("\nEnter the value of root Node: \n");

    scanf("%d", &value);

    fflush(stdin);

    buildtree(arr, 0, value);

    Node(arr);

    printf("\n");

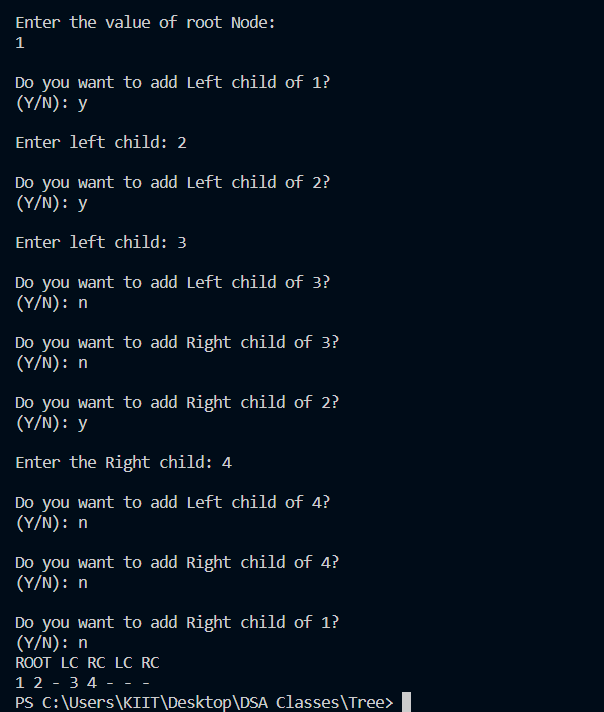
    traversal(arr);

    return 0;

}

//END OF LINE

**Output:**

****

1. **WAP to implement Binary Tree using a linked list and display all the nodes using another function.**

**Input:**

// PROGRAMED BY AKSHAT JAISWAL

#include <stdio.h>

#include <stdlib.h>

*struct* Tree

{

*int* data;

*struct* Tree \*lc;

*struct* Tree \*rc;

};

*void* buildtree(*struct* Tree \**ptr*)

{

*struct* Tree \*New;

*char* ch;

    printf("\nEnter a value: ");

    scanf("%d", &ptr->data);

    fflush(stdin);

    ptr->lc = NULL;

    ptr->rc = NULL;

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

    scanf("%c", &ch);

    fflush(stdin);

    if (ch == 'Y' || ch == 'y')

    {

        New = (*struct* Tree \*)malloc(sizeof(*struct* Tree));

        ptr->lc = New;

        buildtree(New);

    }

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

    scanf("%c", &ch);

    fflush(stdin);

    if (ch == 'Y' || ch == 'y')

    {

        New = (*struct* Tree \*)malloc(sizeof(*struct* Tree));

        ptr->rc = New;

        buildtree(New);

    }

}

*void* INORDER(*struct* Tree \**ptr*)

{

    if (ptr)

    {

        INORDER(ptr->lc);

        printf("%d ", ptr->data);

        INORDER(ptr->rc);

    }

}

*int* main(*void*)

{

*struct* Tree \*Root;

    Root = (*struct* Tree \*)malloc(sizeof(*struct* Tree));

    buildtree(Root);

    printf("\nInorder Traversal:\n");

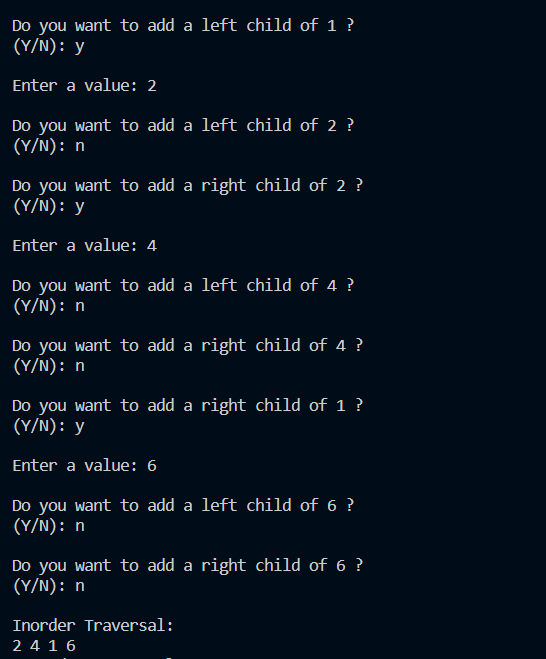
    INORDER(Root);

    return 0;

}

//END OF LINE

**Output:**

****

1. **WAP for inorder, preorder, and postorder traversal using three different functions. Before that, the binary tree is created using a linked list.**

**Input:**

// PROGRAMED BY AKSHAT JAISWAL

#include <stdio.h>

#include <stdlib.h>

*struct* Tree

{

*int* data;

*struct* Tree \*lc;

*struct* Tree \*rc;

};

*void* buildtree(*struct* Tree \**ptr*)

{

*struct* Tree \*New;

*char* ch;

    printf("\nEnter a value: ");

    scanf("%d", &ptr->data);

    fflush(stdin);

    ptr->lc = NULL;

    ptr->rc = NULL;

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

    scanf("%c", &ch);

    fflush(stdin);

    if (ch == 'Y' || ch == 'y')

    {

        New = (*struct* Tree \*)malloc(sizeof(*struct* Tree));

        ptr->lc = New;

        buildtree(New);

    }

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

    scanf("%c", &ch);

    fflush(stdin);

    if (ch == 'Y' || ch == 'y')

    {

        New = (*struct* Tree \*)malloc(sizeof(*struct* Tree));

        ptr->rc = New;

        buildtree(New);

    }

}

*void* INORDER(*struct* Tree \**ptr*)

{

    if (ptr)

    {

        INORDER(ptr->lc);

        printf("%d ", ptr->data);

        INORDER(ptr->rc);

    }

}

*void* PREORDER(*struct* Tree \**ptr*)

{

    if (ptr)

    {

        printf("%d ", ptr->data);

        INORDER(ptr->lc);

        INORDER(ptr->rc);

    }

}

*void* POSTORDER(*struct* Tree \**ptr*)

{

    if (ptr)

    {

        INORDER(ptr->lc);

        INORDER(ptr->rc);

        printf("%d ", ptr->data);

    }

}

*int* main(*void*)

{

*struct* Tree \*Root;

    Root = (*struct* Tree \*)malloc(sizeof(*struct* Tree));

    buildtree(Root);

    printf("\nInorder Traversal:\n");

    INORDER(Root);

    printf("\nPreorder Traversal:\n");

    PREORDER(Root);

    printf("\nPostorder Traversal:\n");

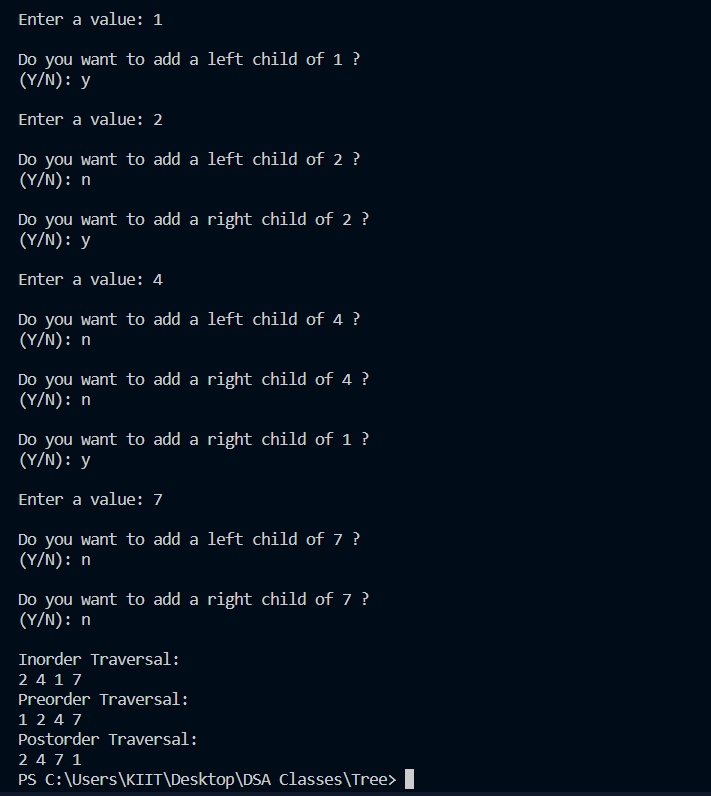
    POSTORDER(Root);

    return 0;

}

//END OF LINE

**Output:**

****