**Name – Akshat Jaiswal**

**Roll No. – 21052646**

**Section – CSE 37**

**DSA LAB 7**

1. **WAP to create a Binary tree and perform the In-order traversal (non-recursive).**

**Input:**

// PROGRAMED BY AKSHAT JAISWAL

#include <stdio.h>

#include <stdlib.h>

*struct* Tree \*STACK[20];

*int* TOP = 0;

*struct* Tree

{

*int* data;

*struct* Tree \*rc;

*struct* Tree \*lc;

};

*void* Inorder(*struct* Tree \**root*)

{

    STACK[TOP] = NULL;

*struct* Tree \*ptr;

    ptr = *root*;

    while (ptr != NULL)

    {

        STACK[++TOP] = ptr;

        ptr = ptr->lc;

    }

    while (TOP != 0)

    {

        ptr = STACK[TOP];

        TOP--;

        printf("%d  ", ptr->data);

        if (ptr->rc != NULL)

        {

            ptr = ptr->rc;

            while (ptr != NULL)

            {

                STACK[++TOP] = ptr;

                ptr = ptr->lc;

            }

        }

    }

}

*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);

    }

}

*int* main(*void*)

{

*struct* Tree \*Root;

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

    buildtree(Root);

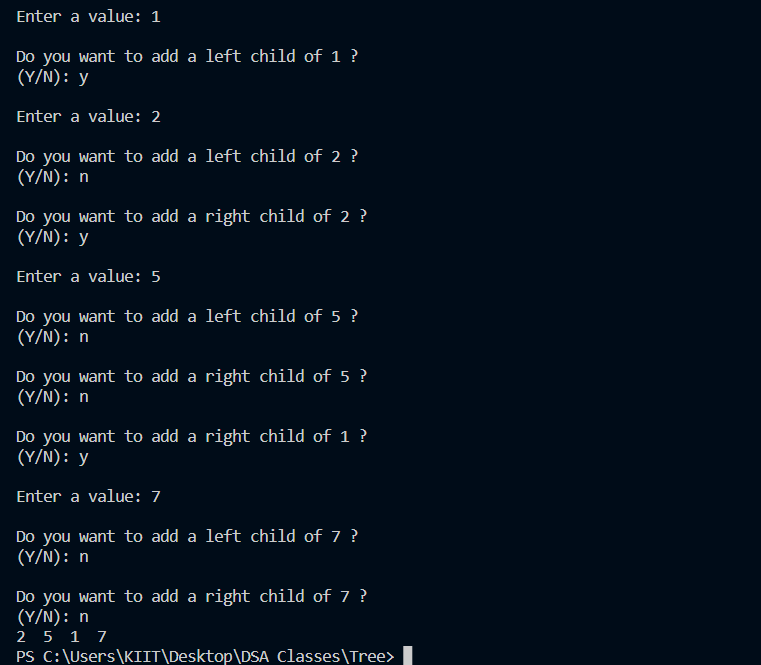
    Inorder(Root);

    return 0;

}

// END OF LINE

**Output:**

****

1. **WAP to create a Binary tree and perform the Pre-order traversal (non-recursive).**

**Input:**

// PROGRAMED BY AKSHAT JAISWAL

#include <stdio.h>

#include <stdlib.h>

*struct* Tree \*STACK[20];

*int* TOP = 0;

*struct* Tree

{

*int* data;

*struct* Tree \*rc;

*struct* Tree \*lc;

};

*void* Preorder(*struct* Tree \**root*)

{

    STACK[TOP] = NULL;

*struct* Tree \*ptr;

    ptr = *root*;

    while (TOP != -1)

    {

        if (ptr == NULL && TOP != 0)

        {

            ptr = STACK[TOP--];

        }

        printf("%d  ", ptr->data);

        if (ptr->rc != NULL)

        {

            STACK[++TOP] = ptr->rc;

            ptr = ptr->lc;

        }

        else if (ptr->lc != NULL)

        {

            ptr = ptr->lc;

        }

        else

        {

            ptr = STACK[TOP];

            TOP--;

        }

    }

}

*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);

    }

}

*int* main(*void*)

{

*struct* Tree \*Root;

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

    buildtree(Root);

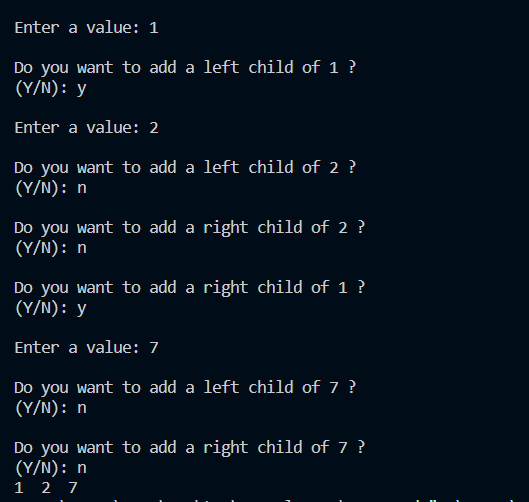
    Preorder(Root);

    return 0;

}

// END OF LINE

**Output:**

****

1. **WAP to create a Binary tree and perform the Post-order traversal (non-recursive).**

**Input:**

// PROGRAMED BY AKSHAT JAISWAL

#include <stdio.h>

#include <stdlib.h>

*struct* Tree \*STACK[20];

*int* TOP = 0;

*struct* Tree

{

*int* data;

*struct* Tree \*rc;

*struct* Tree \*lc;

};

*void* Postorder(*struct* Tree \**root*)

{

*struct* Tree \*ptr, \*temp;

    ptr = *root*;

    STACK[TOP] = NULL;

    do

    {

        while (ptr != NULL)

        {

            if (ptr->rc != NULL)

            {

                STACK[++TOP] = ptr->rc;

            }

            STACK[++TOP] = ptr;

            ptr = ptr->lc;

        }

        ptr = STACK[TOP--];

        if (ptr->rc == STACK[TOP] && ptr->rc != NULL)

        {

            temp = STACK[TOP];

            STACK[TOP] = ptr;

            ptr = temp;

        }

        else

        {

            printf("%d  ", ptr->data);

            ptr = NULL;

        }

    } while (TOP != 0);

}

*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);

    }

}

*int* main(*void*)

{

*struct* Tree \*Root;

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

    buildtree(Root);

    printf("\n[ ");

    Postorder(Root);

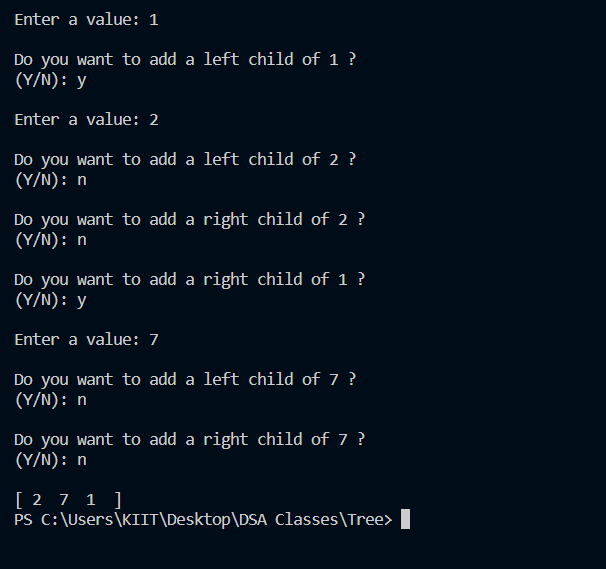
    printf("]");

    return 0;

}

// END OF LINE

**Output:**

****