DSA ASSIGNMENT-6

TANISHA KARMAKAR
21051950
CSE 37

Q1. WAP to implement Binary Tree using array and display all the nodes using another function.

```c
#include<stdio.h>

void buildtree(int t[],int index,int value)
{
    int ch;
    int data;

    t[index]=value;
    printf("Do you have left child of %d (0/1): ",value);
    scanf("%d",&ch);
    if(ch==1)
    {
        printf("Enter the left child: ");
        scanf("%d",&data);
        buildtree(t,2*index+1,data);
    }
    printf("Do you have right child of %d (0/1): ",value);
    scanf("%d",&ch);
    if(ch==1)
    {
        printf("Enter the right child: ");
        scanf("%d",&data);
        buildtree(t,2*index+2,data);
    }

}
int main()
{
    int t[20];
    int index,value;

    for(int i=0;i<20;i++)
    t[i]=-1;

printf("Roll No - 21051950\n");
    printf("Enter the Root Node: ");
    scanf("%d",&value);
    buildtree(t,0,value);
    for(int i=0;i<20;i++)
    {
        if(t[i]==-1)
        printf("_ ");
        else
```

```
        printf("%d ",t[i]);
    }

    return 0;
}
```

OUTPUT:

Roll No - 21051950
Enter the Root Node: 10
Do you have left child of 10 (0/1): 1
Enter the left child: 20
Do you have left child of 20 (0/1): 1
Enter the left child: 30
Do you have left child of 30 (0/1): 0
Do you have right child of 30 (0/1): 1
Enter the right child: 10
Do you have left child of 10 (0/1): 0
Do you have right child of 10 (0/1): 0
Do you have right child of 20 (0/1): 1
Enter the right child: 40
Do you have left child of 40 (0/1): 0
Do you have right child of 40 (0/1): 0
Do you have right child of 10 (0/1): 0
10 20 _ 30 40 _ _ _ 10 _ _ _ _ _ _ _ _ _ _ _

Q.2 WAP to implement Binary Tree using linked list and display all the nodes using another function.

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

struct tree
{
    int data;
    struct tree *lc;
    struct tree *rc;
};
```

```
void buildtree(struct tree *ptr)
{
    int ch;
    printf("enter the value : ");
    scanf("%d",&ptr->data);
    ptr->lc = NULL;
```

```
    ptr->rc = NULL;
    printf("Do you want to add left child of %d (0/1) :", ptr->data);
    scanf("%d",&ch);
```

```
    if (ch==1)
    {
        struct tree *new = (struct tree*)malloc(sizeof(struct tree));
        ptr->lc = new;
        buildtree(new);
    }
```

```
    printf("Do you want to add right child of %d (0/1):", ptr->data);
    scanf("%d",&ch);
```

```
    if (ch==1)
    {
        struct tree *new = (struct tree*)malloc(sizeof(struct tree));
        ptr->rc = new;
        buildtree(new);
    }
}
```

```
void disp(struct tree *ptr)
{
    if(ptr->lc != NULL)
        disp(ptr->lc);
    printf("%d ",ptr->data);
    if(ptr->rc != NULL)
        disp(ptr->rc);
}
```

```
int main()
{
    struct tree *root;
    root = (struct tree*)malloc(sizeof(struct tree));
    buildtree(root);
    disp(root);
}1
```

OUTPUT:

enter the value : 20
Do you want to add left child of 20 (0/1) :1
enter the value : 30
Do you want to add left child of 30 (0/1) :1
enter the value : 40
Do you want to add left child of 40 (0/1) :0
Do you want to add right child of 40 (0/1):1
enter the value : 50
Do you want to add left child of 50 (0/1) :1
enter the value : 30
Do you want to add left child of 30 (0/1) :0

Do you want to add right child of 30 (0/1):0
Do you want to add right child of 50 (0/1):0
Do you want to add right child of 30 (0/1):1
enter the value : 43
Do you want to add left child of 43 (0/1) :0
Do you want to add right child of 43 (0/1):0
Do you want to add right child of 20 (0/1):0
40 30 50 30 43 20

Q.3 WAP for inorder, preorder and postorder traversal using three different functions. Before that the binary tree is created using linked list.

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

struct tree
{
    int data;
    struct tree *lc;
    struct tree *rc;
};
```

```c
void buildtree(struct tree *ptr)
{
    int ch;
    printf("enter the value : ");
    scanf("%d",&ptr->data);
    ptr->lc = NULL;
    ptr->rc = NULL;
    printf("Do you want to add left child of %d (0/1):", ptr->data);
    scanf("%d",&ch);
```

```c
    if (ch==1)
    {
        struct tree *new = (struct tree*)malloc(sizeof(struct tree));
        ptr->lc = new;
        buildtree(new);
    }
```

```c
    printf("Do you want to add right child of %d (0/1) :", ptr->data);
    scanf("%d",&ch);
```

```c
    if (ch==1)
    {
        struct tree *new = (struct tree*)malloc(sizeof(struct tree));
        ptr->rc = new;
        buildtree(new);
    }
}
```

```
void inorder_display(struct tree *ptr)
{

    if(ptr->lc != NULL)
        inorder_display(ptr->lc);
    printf("%d ",ptr->data);
    if(ptr->rc != NULL)
        inorder_display(ptr->rc);
}
void preorder_display(struct tree *ptr)
{

    printf("%d ",ptr->data);
    if(ptr->lc != NULL)
        preorder_display(ptr->lc);
    if(ptr->rc != NULL)
        preorder_display(ptr->rc);
}
void postorder_display(struct tree *ptr)
{

    if(ptr->lc != NULL)
        postorder_display(ptr->lc);
    if(ptr->rc != NULL)
        postorder_display (ptr->rc);
    printf("%d ",ptr->data);
}
```

```
int main()
{
    struct tree *root;
    root = (struct tree*)malloc(sizeof(struct tree));
    buildtree(root);
```

```
    printf("\ninorder traversal : ");
    inorder_display(root);
    printf("\npre-order traversal : ");
    preorder_display(root);
    printf("\npost-order traversal : ");
    postorder_display(root);
}
```

OUTPUT:

enter the value : 10
Do you want to add left child of 10 (0/1):1
enter the value : 20
Do you want to add left child of 20 (0/1):1
enter the value : 30
Do you want to add left child of 30 (0/1):0
Do you want to add right child of 30 (0/1) :1
enter the value : 40
Do you want to add left child of 40 (0/1):

0
Do you want to add right child of 40 (0/1) :1
enter the value : 35
Do you want to add left child of 35 (0/1):0
Do you want to add right child of 35 (0/1) :1
enter the value : 45
Do you want to add left child of 45 (0/1):0
Do you want to add right child of 45 (0/1) :0
Do you want to add right child of 20 (0/1) :0
Do you want to add right child of 10 (0/1) :0

inorder traversal : 30 40 35 45 20 10
pre-order traversal : 10 20 30 40 35 45
post-order traversal : 45 35 40 30 20 10