

# Experiment 6

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

struct node{
int data;
struct node *left;
struct node *right;
};

struct node *tree;
void create(struct node *);
struct node *insert(struct node *, int);
void inorder(struct node *);
void preorder(struct node *);
void postorder(struct node *);

void main()
{
int choice, x;
struct node *ptr;
create(tree);
do
{
printf("\n Operations available are : ");
printf("\n 1. Insert a node");
printf("\n 2. Display inorder traversal");
printf("\n 3. Display preorder traversal");
printf("\n 4. Display postorder traversal");
printf("\n 5. Exit \n");
printf("\n Enter your choice\t");
scanf("%d", &choice);

switch (choice){

case 1:
printf("\n Enter data to be inserted\t");
scanf("%d",&x);
tree = insert(tree, x);
break;

case 2:
printf("\n Elements in the inorder traversal are\t");
inorder(tree);
printf("\n");
break;

case 3:
printf("\n Elements in the preorder traversal are\t");
preorder(tree);
```

```
printf("\n");  
break;
```

```
case 4:  
printf("\n Elements in the postorder traversal are");  
postorder(tree);  
printf("\n");  
break;
```

```
case 5:  
printf("\n Exit: program finished !!!");  
break;  
default:  
printf("\n Please enter a valid option from 1,2,3,4,5. ");  
break;  
}  
}  
while (choice != 5);  
}
```

```
void create(struct node *tree)  
{  
tree = NULL;  
}  
struct node *insert(struct node *tree, int x)  
{  
struct node *p, *temp, *root;  
p = (struct node *)malloc(sizeof(struct node));  
p->data = x;  
p->left = NULL;  
p->right = NULL;  
if (tree == NULL)  
{  
tree = p;  
tree->left = NULL;  
tree->right = NULL;  
}  
else  
{  
root = NULL;  
temp = tree;  
while (temp != NULL)  
{  
root = temp;  
if (x < temp->data)  
temp = temp->left;  
else  
temp = temp->right;  
}  
if(x < root->data)  
root->left = p;
```

```

else
root->right = p;
}
return tree;
}
void inorder(struct node *tree)
{
if (tree != NULL)
{
inorder(tree->left);
printf("%d \t", tree->data);
inorder(tree->right);
}
}

void preorder(struct node *tree){
if (tree != NULL)
{
printf("%d \t", tree->data);
preorder(tree->left);
preorder(tree->right);
}
}

void postorder(struct node *tree){
if (tree != NULL)
{
postorder(tree->left);
postorder(tree->right);
}
}

```

```
printf("%d \t", tree->data);  
}
```

```
dl0410@itadmin:~$ gcc raj6.c  
dl0410@itadmin:~$ ./a.out  
  
Operations available are :  
1. Insert a node  
2. Display inorder traversal  
3. Display preorder traversal  
4. Display postorder traversal  
5. Exit  
  
Enter your choice      1  
  
Enter data to be inserted      12  
  
Operations available are :  
1. Insert a node  
2. Display inorder traversal  
3. Display preorder traversal  
4. Display postorder traversal  
5. Exit  
  
Enter your choice      1  
  
Enter data to be inserted      11  
  
Operations available are :  
1. Insert a node  
2. Display inorder traversal  
3. Display preorder traversal  
4. Display postorder traversal  
5. Exit  
  
Enter your choice      1  
  
Enter data to be inserted      20  
  
Operations available are :  
1. Insert a node  
2. Display inorder traversal
```

3. Display preorder traversal
4. Display postorder traversal
5. Exit

Enter your choice        2

Elements in the inorder traversal are   11        12        20

Operations available are :

1. Insert a node
2. Display inorder traversal
3. Display preorder traversal
4. Display postorder traversal
5. Exit

Enter your choice        3

Elements in the preorder traversal are 12        11        20

Operations available are :

1. Insert a node
2. Display inorder traversal
3. Display preorder traversal
4. Display postorder traversal
5. Exit

Enter your choice        4

Elements in the postorder traversal are 11        20        12

Operations available are :

1. Insert a node
2. Display inorder traversal
3. Display preorder traversal
4. Display postorder traversal
5. Exit

Enter your choice        5

2. Display inorder traversal

}