BST and Traversing

```
#include<stdio.h>
#include<stdlib.h>
typedef struct tree *node; //typedef assigns alternative names to existing data types
node insert(int,node T);
void preorder(node T);
void inorder(node T);
void postorder(node T);
struct tree
{
      int data;
      struct tree *right,*left;
}*root;
int main(void)
{
   node T=NULL;
   int data,ch,i=0,n;
   printf("Enter the no. of elements:");
   scanf("%d",&n);
   printf("\n The elements are:\n");
   while(i<n)
   {
            scanf("%d",&data);
            T=insert(data,T);i++;
   }
```

```
    do
    {
printf("\n1.preorder\t2.inorder\t3.postorder\t4.exit\n");
printf("Enter the choice:");
scanf("%d",&ch);
switch(ch)
{
        case 1:
            printf("Preorder Traversal\n");
            preorder(T);
            break;
        case 2:
            printf("Inorder Traversal\n");
            inorder(T);
            break;
        case 3:
            printf("Postorder Traversal\n");
            postorder(T);
            break;
        case 4:
             exit(0);
}
}while(ch<5);
system("pause");
}
```

```
node insert(int x,node T)
{
    struct tree *newnode;
    newnode=malloc(sizeof(struct tree));
    if(T==NULL)
    {
            newnode->data=x;
            newnode->left=NULL;
            newnode->right=NULL;
            T=newnode;
            }
     else
     {
        if(x<T->data)
        T->left=insert(x,T->left);
        else
        T->right=insert(x,T->right);
        }
        return T;
}
void preorder(node T)
{
    if(T!=NULL)
    {
            printf("%d\t",T->data);
            preorder(T->left);
```

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

void postorder(node T)
{
    if(T!=NULL)
    {
            postorder(T->left);
            postorder(T->right);
            printf("%d\t",T->data);
            }}
```