Program 1

```
computer$ gcc trees.c
computer$ ./a.out
 preorder:  4, 2, 1, 3, 7, 6, 5,
 inorder:  1, 2, 3, 4, 5, 6, 7,
postorder:  1, 3, 2, 5, 6, 7, 4,
```

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

typedef struct node {
	int number;
	struct node *left;
	struct node *right;
} BNODE;


void preorder(BNODE* root)
{
	if(!root) /*root==NULL*/
	{
		return;
	}
		printf(" %d,", root->number);
		preorder(root->left);
		preorder(root->right);
}


void inorder(BNODE* root)
{
	if((!root) /*root==NULL*/
	{
		return;
	}

	inorder(root->left);
	printf(" %d,", root->number);
	inorder(root->right);
}

void postorder(BNODE* root)
{
	if(!root) /*root==NULL*/
	{
		return;
	}
```

```
    postorder(root->left);
    postorder(root->right);
    printf(" %d,", root->number);
}

BNODE* addNode(int number)
{
    BNODE* temp = malloc( sizeof(BNODE) );
    temp->number = number;
    temp->left = NULL;
    temp->right = NULL;

    return temp;
}

void insert(BNODE* root, int number)
{
    if(number <= root->number)
    {
            if(!root->left ) /*root->left==NULL*/
            {
                    root->left = addNode( number );
            }

            else
            {
                    insert(root->left, number);
            }
    }

    else
    {
            if(!root->right) /*root->left==NULL*/
            {
                    root->right = addNode( number );
            }
            else
            {
                    insert(root->right, number);
            }
    }
}



int main(void)
{
    BNODE* root = NULL;
    int i;
    int d[] = {4, 2, 7, 1, 6, 5, 3};

    for(i = 0; i < 7; i++) //note you could put this in a function
    {
        if(!root) /*root==NULL*/
        {
            root = addNode( d[i] );
        }

        else
        {
            insert(root, d[i]);
        }
    }

    printf(" preorder: ");
    preorder(root);
    printf("\n inorder: ");
    inorder(root);
    printf("\npostorder: ");
    postorder(root);
    printf("\n");
}
```