BST 4

```
//node leaves height
#include <stdio.h>
#include <stdlib.h>
int max(int a,int b)
{
	if(a>=b)
		return a;
	return b;
}
int nodes=0,leaves=0;
struct node
{
	int data;
	struct node* left;
	struct node* right;
};
struct node* insert(struct node* root,int x)
{
	if(root == NULL)
	{
		struct node* temp = (node*)malloc(sizeof(struct node));
		temp->data = x;
		temp->left = temp->right = NULL;
		return temp;
	}
	else if(x <= root->data)
		root->left = insert(root->left,x);
	else
		root->right = insert(root->right,x);
	return root;
}
void l_inorder(struct node* root)
{
	if(root == NULL) return;
	l_inorder(root->left);
	if(root->left == NULL && root->right == NULL)
		leaves++;
	l_inorder(root->right);
}
void n_inorder(struct node* root)
{
	if(root == NULL) return;
	n_inorder(root->left);
	nodes++;
	n_inorder(root->right);
}
int height(struct node* root)
{
```

```
	if(root == NULL) return -1;
	else
		return max(height(root->left),height(root->right))+1;
}
int main()
{
	int x,n;
	struct node* root = NULL;
	while(1)
	{
		//printf("1 insert,2 leaves,3 nodes,4 height,5 end\n");
		scanf("%d",&n);
		switch(n)
		{
			case 1:
				scanf("%d",&x);
				root = insert(root,x);
				break;
			case 2:
				leaves = 0;
				l_inorder(root);
				printf("leaves %d\n",leaves);
				break;
			case 3:
				nodes = 0;
				n_inorder(root);
				printf("nodes %d\n",nodes);
				break;
			case 4:
				printf("height %d\n",height(root));
				break;
			case 5:
				return 0;
		}
	}

}
```