```
#include<stdio.h>
#include<stdlib.h>
struct bst
{
int data;
struct bst *lchild,*rchild;
}node;
int cnt=0,leafcnt=0,nleafcnt=0;
struct bst *create()
{
struct bst *temp=(struct bst*)malloc(sizeof(struct bst));
temp->lchild=NULL;
temp->rchild=NULL;
return temp;
}
void insert(struct bst *r, struct bst *new1)
{
if(new1->data < r->data)
{
if(r->lchild==NULL)
r->lchild=new1;
else
insert(r->lchild,new1);
}

if(new1->data > r->data)
{
if(r->rchild==NULL)
r->rchild=new1;
else
insert(r->rchild,new1);
}
}

struct bst *search(struct bst *r, int key)
{
struct bst *temp;
temp=r;
while(temp!=NULL)
{
if(temp->data==key)
return temp;
if(key < temp->data)
temp=temp->lchild;
else
temp=temp->rchild;
}
return NULL;
}

int count(struct bst *temp)
{
if(temp!=NULL)

{
cnt++;
count(temp->lchild);
count(temp->rchild);
}
return cnt;
}
int countleaf(struct bst *temp)
{
if(temp!=NULL)
{
if(temp->lchild==NULL && temp->rchild==NULL)
leafcnt++;
countleaf(temp->lchild);
countleaf(temp->rchild);
}
return leafcnt;
}

int countnleaf(struct bst *temp)
{
if(temp!=NULL)
{
if(temp->lchild!=NULL || temp->rchild!=NULL)
nleafcnt++;
countnleaf(temp->lchild);
countnleaf(temp->rchild);
}
return nleafcnt;
```

```c
}

void inorder(struct bst *temp)
{
if(temp!=NULL)
{
inorder(temp->lchild);
printf("%d\t",temp->data);
inorder(temp->rchild);
}
}

void postorder(struct bst *temp)
{
if(temp!=NULL)
{
postorder(temp->lchild);
postorder(temp->rchild);
printf("%d\t",temp->data);
}
}

void preorder(struct bst *temp)
{
if(temp!=NULL)
{
printf("%d\t",temp->data);
preorder(temp->lchild);
preorder(temp->rchild);

}
}

void inorder_n(struct bst *r)
{
struct bst *stack[100];
int top=-1;
if(r!=NULL)
{
top++;
stack[top]=r;
r=r->lchild;
while(top>=0)
{

while(r!=NULL)
{
top++;
stack[top]=r;
r=r->lchild;
}
r=stack[top];
top--;
printf("%d\t",r->data);
r=r->rchild;

if(r!=NULL)
{
top++;

stack[top]=r;
r=r->lchild;
}
}
}
}
```