* INSERT

void insert(int d){

tree\_node\* t = new tree\_node;

tree\_node\* parent;

t->data = d;

t->left = NULL;

t->right = NULL;

parent = NULL;

if(isEmpty()) root = t;

else{

tree\_node\* curr;

curr = root;

while(curr){

parent = curr;

if(t->data > curr->data)

curr = curr->right;

else curr = curr->left;

}

if(t->data < parent->data) parent->left = t;

else parent->right = t;

}

}

* **PreOrder ( Root – Left – Right )**

void preorder(tree\_node\* p){

if(p != NULL){

cout<<" "<<p->data<<" ";

if(p->left) preorder(p->left);

if(p->right) preorder(p->right);

}

else return;

}

* **InOrder ( Left – Root – Right )**

void inorder(tree\_node\* p){

if(p != NULL){

if(p->left) inorder(p->left);

cout<<" "<<p->data<<" ";

if(p->right) inorder(p->right);

}

else return;

}

* **PostOrder ( Left – Right – Root )**

void postorder(tree\_node\* p){

if(p != NULL){

if(p->left) postorder(p->left);

if(p->right) postorder(p->right);

cout<<" "<<p->data<<" ";

}

else return;

}

* **Find Value**

bool find (tree\_node\* p, int value){

if(p!=NULL){

if(value==p->data)

return true;

else if(value < p->data){

p=p->left;

find(p,value);

return find(p,value);

}

else if(value > p->data){

p=p->right;

find(p,value);

return(p,value);

}

}

else

return false;

}

* Root dengan 1 child

void satu(tree\_node\* p){

if(p != NULL){

if( (p->left!=NULL && p->right==NULL) || (p->left==NULL && p->right!=NULL) ){

cout<<" "<<p->data<<" ";

}

if(p->left) satu(p->left);

if(p->right) satu(p->right);

}

else return;

}