postorder

#include <iostream>

#include <string>

using namespace std;

void post(const Tree& T){

stack <node&> s;

stack <node&> q;

s.push(T.root)

while(!s.empty()){

node crr = s.top()

if(isExternal(crr)){

cout<<crr.value;

s.pop();

q.push(crr);

}

else if (crr.leftnode==q.top()||crr.rightnode==q.top()){

node& safe= q.top()

q.pop();

if (crr.leftnode==q.top()||crr.rightnode==q.top()){

q.pop()

cout<<crr.value;

s.pop();

q.push(crr);

}

else if (rightnode ==null || lefttnode==null){

cout<<crr.value;

s.pop();

q.push(crr);

}

else q.push(safe);

}

else { if(crr.leftnode != null) s.push(crr.leftnode);

if(crr.leftnode != null) s.push(curr.rightnode);

}

}

}

int main()

{

return 0;

}

Preorder

#include <iostream>

#include <string>

using namespace std;

void post(const Tree& T){

stack <node&> s;

s.push(T.root)

while(!s.empty()){

node crr = s.top()

cout<<crr.value;

s.pop();

if(crr.leftnode != null)s.push(crr.leftnode);

if(crr.leftnode != null)s.push(curr.rightnode);

}

}

int main()

{

return 0;

}

BF tr

#include <iostream>

#include <string>

using namespace std;

void post(const Tree& T){

queue <node&> s;

s.push(T.root)

while(!s.empty()){

node crr = s.front()

cout<<crr.value;

s.pop();

if(crr.leftnode != null)s.push(crr.leftnode);

if(crr.leftnode != null)s.push(curr.rightnode);

}

}

int main()

{

return 0;

}

#include <iostream>

#include <string>

using namespace std;

void bbsh(const Tree& T){

stack <node&> s;

s.push(T.root);

while(!s.empty()){

int crr = s.top;

s.pop;

cout<<crr;

if(crr.rightchild != null){

s.push(crr.rightchlid);

}

if(crr.leftchild != null){

s.push(crr.leftchild);

}

}

int main(){

return 0;

}