TREE

#include <cstdio>

#include <cstdlib>

#include <iostream>

using namespace std;

struct node{

int index,sum,mn,mx,num;

struct node \*left,\*right;

}\*root;

int N;

int \*ar;

void update( node \*p ){

if( p->left && p->right ){

p->num=1+p->left->num+p->right->num;

p->sum=p->index+p->left->sum+p->right->sum;

p->mn=min(p->index,min(p->left->mn,p->right->mn));

p->mx=max(p->index,max(p->left->mx,p->right->mx));

}

else if( p->left && !p->right ){

p->num=1+p->left->num;

p->sum=p->index+p->left->sum;

p->mn=min(p->index,p->left->mn);

p->mx=max(p->index,p->left->mx);

}

}

void add( node \*p , int k ){

if( !p->left ){

p->left=new node;

p->left->index=p->left->sum=p->left->mn=p->left->mx=k;

p->left->num=1;

p->left->left=NULL;

p->left->right=NULL;

}

else if( !p->right ){

p->right=new node;

p->right->index=p->right->sum=p->right->mn=p->right->mx=k;

p->right->num=1;

p->right->left=NULL;

p->right->right=NULL;

}

else if( p->left->num==p->right->num )

add(p->left,k);

else

add(p->right,k);

update(p);

}

void createTree(){

cout << "Enter N: ";

cin >> N;

root=new node;

ar=new int[N];

for( int i=0 ; i<N ; i++)

ar[i]=rand()%N;

root->index=root->mn=root->mx=ar[0];

root->num=1;

for( int i=1 ; i<N ; i++ )

add(root,ar[i]);

for( int i=0 ; i<N ; i++ )

cout << ar[i] << ' ';

cout << endl;

delete(ar);

}

void printPreorder( node \*p ){

printf("%d ",p->index );

if( p->left )

printPreorder(p->left);

if( p->right )

printPreorder(p->right);

}

void printInorder( node \*p ){

if( p->left )

printInorder(p->left);

printf("%d ",p->index );

if( p->right )

printInorder(p->right);

}

void printPostorder( node \*p ){

if( p->left )

printPostorder(p->left);

if( p->right )

printPostorder(p->right);

printf("%d ",p->index );

}

void removeTree( node \*p ){

if( p->left )

removeTree(p->left);

if( p->right )

removeTree(p->right);

p->left=NULL;

p->right=NULL;

delete(p);

}

int findMax(){

return root->mx;

}

int findMin(){

return root->mn;

}

int findNumNode(){

return root->num;

}

int findNumLeaf( node \*p ){

if( p->left && p->right )

return findNumLeaf(p->left)+findNumLeaf(p->right);

if( p->left && !p->right )

return findNumLeaf(p->left);

return 1;

}

int calculateDepth( node \*p , int dep ){

if( p->left && p->right )

return max(calculateDepth(p->left,dep+1),calculateDepth(p->right,dep+1));

if( p->left && !p->right )

return calculateDepth(p->left,dep+1);

return dep;

}

int calculateSum(){

return root->sum;

}

double calcualateAverage(){

return (double)root->sum/(double)N;

}

int main(){

srand(time(NULL));

createTree();

string s;

while(22){

cin.ignore(1000,'\n');

cin >> s;

if( s=="create" )

createTree();

else if( s=="remove" ){

removeTree(root);

root=NULL;

}

else if( s=="pre" ){

printPreorder(root);

cout << endl;

}

else if( s=="in" ){

printInorder(root);

cout << endl;

}

else if( s=="post" ){

printPostorder(root);

cout << endl;

}

else if( s=="max" )

cout << "Maximum value of tree is: " << findMax() << endl;

else if( s=="min" )

cout << "Minimum value of tree is: " << findMin() << endl;

else if( s=="num" )

cout << "Number of nodes in the tree is: " << findNumNode() << endl;

else if( s=="leaf" )

cout << "Number of leaves in the tree is: " << findNumLeaf(root) << endl;

else if( s=="depth" )

cout << "Depth of the tree is: " << calculateDepth(root,1) << endl;

else if( s=="sum" )

cout << "Sum of all values in the tree is: " << calculateSum() << endl;

else if( s=="average" )

cout << "Average of all values in the tree is: " << calcualateAverage() << endl;

else if( s=="exit" )

break;

else

cout << "Invalid choice" << endl;

}

if( root )

removeTree(root);

return 0;

}