[**//1.11/**](#_izb3x99j2xpy) **2**

[//2.11/](#_jywvh3s65vhu) 5

[//3.11/](#_h518knijenat) 7

[//4.2/](#_iddh1f23k6fk) 10

# //1.11/

#include <iostream>

#include <math.h>

using namespace std;

struct node{

int data;

node\* prev, \*next;

};

void addListLast(node\*\* head, node\*\* tail, int data)

{

node\* temp;

if (!\*head)

{

temp = new node;

temp->data = data;

temp->next = temp->prev = temp;

\*head = \*tail = temp;

}

else

{

temp = new node;

temp->data = data;

temp->next = \*tail;

(\*tail)->prev = temp;

temp->prev = \*head;

(\*head)->next = temp;

\*tail = temp;

}

}

void showList(node \*head, node \*tail)

{

if (!head)

return;

node\* temp = head;

int i = 1;

while (temp)

{

cout << i <<". " << temp->data <<endl;

if (temp->prev != head)

temp = temp->prev;

else temp = NULL;

i++;

}

}

void sumList(node\* head, int& sum, int& height)

{

if (!head)

return;

node\* temp = head;

sum = 0;

do

{

height++;

sum += temp->data;

temp = temp->prev;

} while (temp != head);

}

void findNumbersGreaterThanV(node\* head, double ser, int V)

{

if (!head)

{

cout <<endl<<endl<< "this tree does not exist";

return;

}

node\* temp = head;

bool found = false;

do

{

if (((temp->data - ser < ser) and (temp->data - ser <= V) and (temp->data - ser > 0)) or ((ser - temp->data < ser) && (ser - temp->data <= V) && (ser - temp->data > 0) ) or (ser == temp->data))

{

cout << temp->data << " ";

found = true;

}

temp = temp->prev;

} while (temp != head);

if (!found)

cout << "No numbers greater than " << ser << " found";

cout << endl;

}

int main()

{

node\* head, \*tail;

head = tail = NULL;

for (int i = 0; i < 10; i++)

addListLast(&head, &tail, i);

showList(head, tail);

int height = 0;

int sum = 0;

sumList(head, sum, height);

// cout << "Sum of list elements: " << sum<< "height: " << height << endl;

cout << "Sum: " << sum<<endl<< "height: " << height << endl;

double ser = static\_cast<double>(sum) / height;

cout << "ser " << ser << ": ";

cout << "Numbers greater than " << ser << ": ";

int V = 2;

findNumbersGreaterThanV(head, ser, V);

return 0;

}

## //2.11/

#include <iostream>

using namespace std;

struct node{

int num;

node\* left;

node\* right;

};

void addNode(node\* nod, int data){

if (data < nod->num)

if (nod->left)

addNode(nod->left, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->left = newNode;

}

else

if (nod->right)

addNode(nod->right, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->right = newNode;

}

}

void TREEHEIGHT(node\* nod, int \*check, int height){

if (!nod)

return;

height++;

cout<<"data/height = "<<nod->num<< " / "<< height<<endl;

if (\*check<height)

\*check=height;

if (nod->left)

TREEHEIGHT(nod->left, check, height);

if (nod->right )

TREEHEIGHT(nod->right, check, height);

}

int main(){

int arr[14] = {9,14,5,7,13,16,3,12,20,4,11,6,15,16};

node\* root = new node;

root->left = root->right = NULL;

root->num = 10;

for (int i = 0; i < 14; i++)

addNode(root, arr[i]);

int height = 0;

int check = 0;

TREEHEIGHT(root, &check, height);

cout<< endl<<"MAXheight = "<< check;

return 0;

}

### //3.11/

#include <iostream>

using namespace std;

struct node{

int num;

node\* next;

node(){

next = NULL;

}

};

struct graph{

node\* arr[10];

graph()

{

for (int i = 0; i < 10; i++)

arr[i] = NULL;

}

};

graph matrixToGraph(int mtr[10][10]){

graph graph1;

node\* tempLast;

for (int i = 0; i < 10; i++)

for (int j = 0; j < 10; j++)

if (mtr[i][j] != 0)

{

node\* temp;

temp = new node;

temp->num = j+1;

if (!graph1.arr[i])

graph1.arr[i] = temp;

else

tempLast->next = temp;

tempLast = temp;

}

return graph1;

}

void showMatrix(int arr[10][10], int max){

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

{

for (int j = 0; j < max; j++)

cout << arr[i][j] << "; ";

cout << endl;

}

}

void showStruct(graph graph1, int max){

cout << "struct:" << endl;

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

{

node\* temp = graph1.arr[i];

cout << "node: " << i+1 << " -> ";

while (temp)

{

cout << temp->num << "; ";

temp = temp->next;

}

cout << endl;

}

}

int counter(graph graph1, int max){

int counter = 0;

for (int i = 0; i < max; i++){

node\* temp = graph1.arr[i];

if (!temp)

counter++;

}

return counter;

}

int main(int argc, char\*\* argv) {

int mtrx1[10][10] = {

{0,1,1,0,1,0,0,0,0},

{1,0,0,0,1,0,0,0,0},

{1,0,0,1,0,0,1,0,0},

{0,0,1,0,0,0,0,0,0},

{1,1,0,0,0,1,1,0,0},

{0,0,0,0,1,0,0,0,0},

{0,0,1,0,1,0,0,0,0},

{0,0,0,0,0,0,0,0,0},

{0,0,0,0,0,0,0,0,0}

};

graph graph1;

graph1 = matrixToGraph(mtrx1);

showMatrix(mtrx1, 10);

showStruct(graph1, 10);

cout << "Number of isolated nodes: " << counter(graph1, 10) << endl;

return 0;

}

#### //4.2/

#include <iostream>

using namespace std;

struct node{

int num;

node\* left;

node\* right;

};

void addNode(node\* nod, int data){

if (data < nod->num)

if (nod->left)

addNode(nod->left, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->left = newNode;

}

else

if (nod->right)

addNode(nod->right, data);

else

{

node\* newNode = new node;

newNode->num = data;

newNode->left = newNode->right = NULL;

nod->right = newNode;

}

}

void showBinary(node\* nod){

if (!nod)

return;

showBinary(nod->left);

cout << nod->num<<"; ";

showBinary(nod->right);

}

void deleteNode(node\*\* root, int key)

{

node\* del = \*root, \*delParent = NULL;

if (del->num == key)

{ cout <<"deletenum = "<<del->num<<endl;

if (!del)

{

cout << "no such elemet!";

return;

}

// search for change:

if (del->left and del->right)

{

node\* change, \*changeParent;

change = del->right;

changeParent = del;

while (change->left)

{

changeParent = change;

change = change->left;

}

cout << "change found! "<< change->num<<endl;

cout << "parent change found! "<< changeParent->num<<endl;

del->num = change->num;

if (del == changeParent)

changeParent->right = change->right;

else

changeParent->left = change->right;

delete change;

}

else if (del->left)

{

\*root = del->left;

delete del;

}

else if (del->right)

{

\*root = del->right;

delete del;

}

else {

\*root = NULL;

cout << "BIGNULL";

delete del;

}

}

else

{

cout <<"deletenum = "<<key<<endl;

while (del and del->num != key)

{

if (del->num > key)

{

delParent = del;

del = del->left;

}

else

{

delParent = del;

del = del->right;

}

}

if (!del)

{

cout << "no such elemet!"<<endl;

return;

}

else

{

cout << "parent element found! "<< delParent->num<<endl;

}

/// for leaf

if (!del->left and !del->right)

{

if (del == delParent->left)

delParent->left = NULL;

else

delParent->right = NULL;

delete del;

return;

}

// for only 1 offspring

if (del->left and !del->right)

{

if (del == delParent->left)

delParent->left = del->left;

else

delParent->right = del->left;

delete del;

return;

}

if (!del->left and del->right)

{

if (del == delParent->left)

delParent->left = del->right;

else

delParent->right = del->right;

delete del;

return;

}

// search for change:

node\* change, \*changeParent;

change = del->right;

changeParent = del;

while (change->left)

{

changeParent = change;

change = change->left;

}

cout << "change found! "<< change->num<<endl;

cout << "parent change found! "<< changeParent->num<<endl;

del->num = change->num;

if (del == changeParent)

changeParent->right = change->right;

else

changeParent->left = change->right;

delete change;

}

}

int main(){

int arr[14] = {9,14,5,7,13,17,3,12,20,4,11,6, 15, 16};

node\* root = new node;

root->left = root->right = NULL;

root->num = 10;

for (int i = 0; i < 14; i++)

addNode(root, arr[i]);

showBinary(root);

cout << endl<<endl;

int delnum;

// element to be remove

cout << "enter the element you want to remove"<<endl;

cin>> delnum;

cout << "delete:" <<endl;

deleteNode(&root, delnum);

showBinary(root);

return 0;

}