Akshay G

(BL.EN.U4CSE14005)

//STAR

#include <iostream>

using namespace std;

int nodeId = 1;

class Node{

public:

Node \*lptr,\*rptr;

int id;

int visited;

char data;

int lvalue,rvalue;

Node(){

lvalue = -1;

rvalue = -1;

lptr = NULL;

rptr = NULL;

id = nodeId++;

visited = 0;

} };

class List{

public:

Node \*head;

List(){ head = NULL; }

void createNode(char info){

Node \*temp = new Node();

temp->data = info;

head = temp;

Node \*temp2 = new Node();

temp2->data = '$';

temp->rptr = temp2;

temp->rvalue = temp2->id; }

void star(){

Node \*temp1 = new Node();

temp1->data = '#';

Node \*temp2 = new Node();

temp2->data = '$';

Node \*cur = head;

temp1->lptr = cur;

temp1->lvalue = cur->id;

temp1->rptr = temp2;

temp1->rvalue = temp2->id;

while(cur->rptr!=NULL) cur = cur->rptr;

cur->data = '#';

cur->rptr = temp2;

cur->rvalue = temp2->id;

cur->lptr = head;

cur->lvalue = head->id;

head = temp1;

}

void display(){

BFS(head);

initVisited(head);

}

void BFS(Node \*h){

if(h==NULL) return;

if(h->visited!=1)

cout<<h->data<<" "<<h->id<<" "<<h->lvalue<<" "<<h->rvalue<<endl;

else return;

h->visited = 1;

BFS(h->lptr);

BFS(h->rptr);

}

void initVisited(Node \*h){

if(h == NULL)

return;

if(h->visited == 0)

return;

h->visited = 0;

initVisited(h->lptr);

initVisited(h->rptr);

}

};

int main(){

List l1;

l1.createNode('a');

l1.star();

l1.display();

}

OUTPUT:

# 3 1 4

a 1 -1 2

# 2 1 4

$ 4 -1 -1

Akshay G

(BL.EN.U4CSE14005)

//DOT

#include <iostream>

using namespace std;

int nodeId = 1;

class Node{

public:

Node \*lptr,\*rptr;

int id;

int visited;

char data;

int lvalue,rvalue;

Node(){

lvalue = -1;

rvalue = -1;

lptr = NULL;

rptr = NULL;

id = nodeId++;

visited = 0;

} };

class List{

public:

Node \*head;

List(){ head = NULL; }

void createNode(char info){

Node \*temp = new Node();

temp->data = info;

head = temp;

Node \*temp2 = new Node();

temp2->data = '$';

temp->rptr = temp2;

temp->rvalue = temp2->id; }

void dot(List l){

Node \*temp1 = head;

Node \*temp2 = l.head;

while(temp1->rptr->data!='$')

temp1 = temp1->rptr;

temp1->rptr = temp2;

temp1->rvalue = temp2->id;

temp1 = head;

if(temp1->lptr != NULL){

temp1 = temp1->lptr;

while(temp1->rptr->data!='$')

temp1 = temp1->rptr;

temp1->rptr = temp2;

temp1->rvalue = temp2->id;

}

}

void display(){

BFS(head);

initVisited(head);

}

void BFS(Node \*h){

if(h==NULL) return;

if(h->visited!=1)

cout<<h->data<<" "<<h->id<<" "<<h->lvalue<<" "<<h->rvalue<<endl;

else return;

h->visited = 1;

BFS(h->lptr);

BFS(h->rptr);

}

void initVisited(Node \*h){

if(h == NULL)

return;

if(h->visited == 0)

return;

h->visited = 0;

initVisited(h->lptr);

initVisited(h->rptr);

}

};

int main(){

List l1,l3;

l1.createNode('a');

l3.createNode('b');

l1.dot(l3);

l1.display();

}

OUTPUT:

a 1 -1 3

b 3 -1 4

$ 4 -1 -1

Akshay G

(BL.EN.U4CSE14005)

//PLUS

#include <iostream>

using namespace std;

int nodeId = 1;

class Node{

public:

Node \*lptr,\*rptr;

int id;

int visited;

char data;

int lvalue,rvalue;

Node(){

lvalue = -1;

rvalue = -1;

lptr = NULL;

rptr = NULL;

id = nodeId++;

visited = 0;

} };

class List{

public:

Node \*head;

List(){ head = NULL; }

void createNode(char info){

Node \*temp = new Node();

temp->data = info;

head = temp;

Node \*temp2 = new Node();

temp2->data = '$';

temp->rptr = temp2;

temp->rvalue = temp2->id; }

void plus(List l){

Node \*l1 = head;

Node \*l2 = l.head;

Node \*temp1 = new Node();

temp1->data = '#';

temp1->lptr = head;

temp1->lvalue = l1->id;

temp1->rptr = l2;

temp1->rvalue = l2->id;

Node \*temp2 = new Node();

temp2->data = '$';

while(l1->data!='$') l1 = l1->rptr;

while(l2->data!='$') l2 = l2->rptr;

l1->data = '#';

l2->data = '#';

l1->rptr = temp2;

l1->rvalue = temp2->id;

l2->rptr = temp2;

l2->rvalue = temp2->id;

head = temp1;

}

void display(){

BFS(head);

initVisited(head);

}

void BFS(Node \*h){

if(h==NULL) return;

if(h->visited!=1)

cout<<h->data<<" "<<h->id<<" "<<h->lvalue<<" "<<h->rvalue<<endl;

else return;

h->visited = 1;

BFS(h->lptr);

BFS(h->rptr);

}

void initVisited(Node \*h){

if(h == NULL)

return;

if(h->visited == 0)

return;

h->visited = 0;

initVisited(h->lptr);

initVisited(h->rptr);

}

};

int main(){

List l1,l3;

l1.createNode('a');

l3.createNode('b');

l1.plus(l3);

l1.display();

}

OUTPUT:

# 7 1 3

a 1 -1 2

# 2 -1 8

$ 8 -1 -1

b 3 -1 4

# 4 -1 8