$ cat head

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* \*

\* Лабораторная работа №23 \*

\* \*

\* Динамические структуры данных \*

\* Обработка деревьев \*

\* \*

\* Работу выполнил: \*

\* Стеснягин Семён \*

\* Группа: 80-102Б \*

\* \*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$ cat lab23\_var14.c

#include <stdio.h>

#include <stdlib.h>

#include "tree.h"

tree \*insert(tree \*treep, tree \*newp);

void printree(tree \*t);

tree \*minimum(tree \*x);

tree \*delete(tree \*root, char z);

int traverse\_tree(tree \*l, int level, int \*result, int \*ans);

int isLeaf(tree \*l);

int main(void) {

int exit = 0;

tree \*tr = NULL;

char action;

while ((scanf("\n%c", &action)) != 0) {

char input;

if (action == '+') {

scanf("\n%c", &input);

tree \*newp = (tree \*) malloc(sizeof(tree));

newp->val = input;

newp->left = NULL;

newp->right = NULL;

tr = insert(tr, newp);

}

else if (action == 'p') {

printree(tr);

}

else if (action == '-') {

scanf("\n%c", &input);

tr = delete(tr, input);

}

else if (action == 'f') {

int a = 0;

int ans = 1;

if (tr != NULL) {

if (traverse\_tree(tr, 0, &a, &ans) == 1) {

printf("All leaves have the same level\n");

}

else {

printf("Leaves have different level\n");

}

}

}

}

}

tree \*insert(tree \*treep, tree \*newp) {

if (treep == NULL) {

return newp;

}

if (treep->val == newp->val) {

printf("Duplicate ignored\n");

}

else if (treep->val > newp->val) {

treep->left = insert(treep->left, newp);

} else {

treep->right = insert(treep->right, newp);

}

return treep;

}

void rRl(tree \*t, int l) {

if (t != NULL) {

rRl(t->right, l + 1);

printf("%\*s%c\n", 4 \* l, " ", t->val);

rRl(t->left, l + 1);

}

}

void printree(tree \*t) {

rRl(t, 0);

printf("\n");

}

tree \*delete(tree \*root, char z) {

if (root == NULL)

return root;

if (z < root->val) {

root->left = delete(root->left, z);

}

else if (z > root->val) {

root->right = delete(root->right, z);

}

else if (root->left != NULL && root->right != NULL) {

root->val = minimum(root->right)->val;

root->right = delete(root->right, root->right->val);

}

else {

if (root->left != NULL)

root = root->left;

else {

root = root->right;

}

}

return root;

}

tree \*minimum(tree \*x) {

if (x->left == NULL)

return x;

return minimum(x->left);

}

int traverse\_tree(tree \*l, int level, int \*result, int \*ans) {

if (l != NULL) {

level++;

traverse\_tree(l->left, level, result, ans);

if (isLeaf(l)) {

if (\*result == 0) {

\*result = level;

}

else if (\*result != level) {

return \*ans = 0;

}

}

traverse\_tree(l->right, level, result, ans);

}

return \*ans;

}

int isLeaf(tree \*l) {

return (l->right == NULL) && (l->left == NULL);

}

$ cat tree.h

typedef struct tree tree;

struct tree {

char val;

tree \*left;

tree \*right;

};

$ cc lab23\_var14.c

$ ./a.out

+ a

+ b

+ c

+ d

+ 1

+ 0

+ 2

p

d

c

b

a

2

1

0

f

Leaves have different level

- d

p

c

b

a

2

1

0

f

All leaves have the same level

- b

p

c

a

2

1

0

- 1

p

c

a

2

0

+ c

Duplicate ignored

- 1

p

c

a

2

0