#include <iostream>

#include <string>

using namespace std;

struct Node //звено дерева

{

string info;

Node \*left, \*right; //указатели на новые звенья

};

Node \*root = NULL;

Node\* List(string word)

{

Node \*list = new Node;

list->info = word;

list->left = list->right = NULL;

return list;

}

void NewList(string slovo)

{

Node \*papka, \*temp;

int search = 1;

papka = NULL;

temp = root;

while (temp && search)

{

papka = temp;

if (slovo == temp->info)

{

search = 0;

cout << "Item already exists!" << endl;

return;

}

else

{

if (slovo < temp->info)

temp = temp->left;

else

temp = temp->right;

}

}

if (search)

temp = List(slovo);

if (slovo < papka->info)

papka->left = temp;

if (slovo > papka->info)

papka->right = temp;

}

void Show(Node \*aaa, int level)

{

if (root == NULL)

{

cout << "Tree is empty ... " << endl;

system("pause");

return;

}

if (aaa)

{

Show(aaa->left, level + 1); //вывод левого поддерева

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

cout << "#";

cout << aaa->info << endl; //выводим корень

Show(aaa->right, level + 1); //вывод правого поддерева

}

}

int N = 0;

void IndTask(Node \*aaa)

{

if (root == NULL)

{

cout << "Tree is empty ... " << endl;

system("pause");

return;

}

if (aaa)

{

IndTask(aaa->right);

if (aaa->info[0] == 'a')

{

N++;

cout << aaa->info << endl;

}

IndTask(aaa->left);

}

}

void Del(Node \*&root) {

if (root != NULL) //Пока не встретится пустое звено

{

Del(root->left); //Рекурсивная функция прохода по левому поддереву

Del(root->right); //Рекурсивная функци для прохода по правому поддереву

delete root; //Убиваем конечный элемент дерева

}

}

int main()

{

string slovo;

int wh = 0, oper, n;

while (true)

{

cout << "1 - Create/Add\n2 - View\n3 - Delete\n4 - Individual Task\n0 - Exit\n => ";

cin >> oper;

switch (oper)

{

case 1:

cout << "Enter the number of elements : ";

cin >> n;

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

{

cout << "Input word : ";

cin >> slovo;

if (root == NULL)

root = List(slovo);

else

NewList(slovo);

}

break;

case 2:

Show(root, 0);

cout << endl;

break;

case 3:

Del(root);

break;

case 4:

IndTask(root);

cout << "N = " << N << endl;

N = 0;

system("pause");

break;

case 0:

return 0;

}

}

system("pause");

}