/\* -----------------------------------------------

Name: J-Zach Loke

Course: CMPS-385

Semester: Spring 2020

Project: Exam 2

Purpose: Store an array of president names in a BST and

a. display the BST to see the names in alphabeitcal order

b. count and display the number of leaves in the tree

c. which presidents have left and right children

----------------------------------------------- \*/

#include <iostream>

#include <string>

class BST

{

private:

struct node

{

std::string info;

node \*left, \*right;

};

node \*root;

void place(node \*insert, node \*curr)

{

if (insert->info < curr->info)

{

if (curr->left == NULL) { curr->left = insert; }

else { place(insert, curr->left); }

}

else if (insert->info > curr->info)

{

if (curr->right == NULL) { curr->right = insert;}

else { place(insert, curr->right); }

}

}

public:

BST()

{

root = NULL;

}

node\* getRoot()

{

return root;

}

void insert(std::string x)

{

node \*insert = new node;

insert->info = x;

insert->left = NULL;

insert->right = NULL;

if (root == NULL) { root = insert; }

else { place(insert, root); }

}

void displayPre(node \*curr)

{

if (curr != NULL)

{

std::cout << curr->info << '\t';

displayPre(curr->left);

displayPre(curr->right);

}

}

void displayIn(node \*curr)

{

if (curr != NULL)

{

displayIn(curr->left);

std::cout << curr->info << '\t';

displayIn(curr->right);

}

}

void displayPost(node \*curr)

{

if (curr != NULL)

{

displayPost(curr->left);

displayPost(curr->right);

std::cout << curr->info << '\t';

}

}

int getLeafCount(node \*curr)

{

if (curr == NULL) { return 0; }

else if (curr->left == NULL && curr->right == NULL) { return 1; }

else

{

return getLeafCount(curr->left) + getLeafCount(curr->right);

}

}

int getParentCount(node \*curr)

{

if (curr == NULL) { return 0; }

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

{

return 1 + getParentCount(curr->left) + getParentCount(curr->right);

}

else

{

return getParentCount(curr->left) + getParentCount(curr->right);

}

}

};

int main()

{

/\* name: main

input: N/A

output: N/A

purpose: main function to drive the program \*/

// given

std::string presidents[10] = { "Johnson Lyndon",

"Nixon Richard",

"Ford Gerald",

"Carter Jimmy",

"Reagan Ronald",

"Bush H. George",

"Clinton Bill",

"Bush W. George",

"Obama Barack",

"Trump Donald" };

BST tree = BST();

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

{

tree.insert(presidents[i]);

}

// part a

tree.displayIn(tree.getRoot());

std::cout << std::endl;

// part b

std::cout << "# of leaves: " << tree.getLeafCount(tree.getRoot()) << std::endl;

// part c

std::cout << "# of presidents with left and right children: " << tree.getParentCount(tree.getRoot()) << std::endl;

system("pause");

return 0;

}

