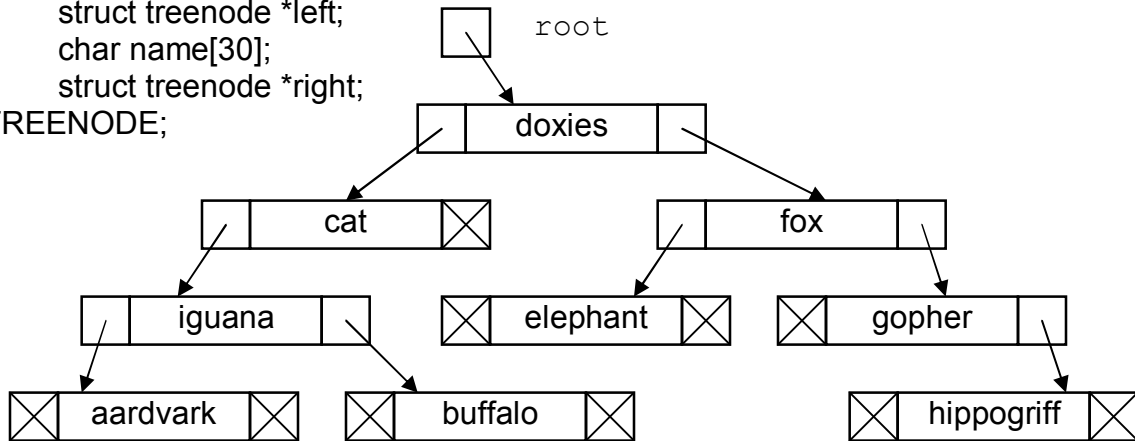


ECE 161
Project 9

Name: tree.cpp
Due: see <http://ece161.viall.org>

Attached is a program which creates and prints the tree below. It also reads and writes this tree structure from or to a file.

```
typedef struct treenode
{
    struct treenode *left;
    char name[30];
    struct treenode *right;
} TREENODE;
```



Write a program which processes the following commands:

Print (or P) – prints the tree

addLeft (or L) – prompt the user for an existing_animal and a new_animal. The new_animal is added to the left node of the existing_animal. If the existing_animal doesn't exist, or the left node of the existing_animal is not null, then the tree is unchanged, and an error message is displayed.

addriGht (or G) – same as "addleft", except adds to the right node.

Search (or S) – prompt the user for an animal; program will search tree and inform user that the the animal does or does not exists.

Read (or R) –reads the tree from "mytree.txt"

Write (or W) – writes the tree to "mytree.txt"

Note: For debugging purposes, save a copy of "mytree.txt" under a different name. That way if your program writes garbage to the file, you can easily fix it.

```

// ECE 161 - Example of reading/writing tree structure to file
// P. Viall Spring 2013
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

typedef struct treenode
{
    char name[20];
    struct treenode *left;
    struct treenode *right;
} TREENODE;

TREENODE * CreateNode(char n[]);
void AddLeft(TREENODE *p, char n[]);
void AddRight(TREENODE *p, char n[]);
void PrintTree(TREENODE *r, int numspace);
void WriteFileDetail(TREENODE *r, FILE *f);
void WriteFile(TREENODE *p);
void ReadFileDetail(TREENODE *r, FILE *f);
void ReadFile(TREENODE **p);

void main(void)
{
    TREENODE *tmp;
    TREENODE *root;
    char line[20];
    printf("Type \"init\" to initialize the tree, else hit ENTER");
    gets(line);

    if (!strcmp("init", line))
    {
        root = CreateNode("doxies");
        tmp = root;          AddLeft(root, "cat");          AddRight(root, "fox");
        tmp = root->left;    AddLeft(tmp, "iguana");
        tmp = root->right;   AddLeft(tmp, "elephant");     AddRight(tmp, "gopher");
        tmp=root->left->left; AddLeft(tmp, "aardvark");     AddRight(tmp, "buffalo");
        tmp=root->right->right; AddRight(tmp, "hippogriff");
        PrintTree(root, 0);
        WriteFile(root);
    }
    ReadFile(&root);
    PrintTree(root, 0);
}

TREENODE * CreateNode(char n[])
{
    TREENODE *p;
    p = (TREENODE *) malloc(sizeof(TREENODE));
    strcpy(p->name, n);
    p->left = p->right = NULL;
    return p;
}

void AddLeft(TREENODE *p, char n[])
{
    p->left = CreateNode(n);
}

void AddRight(TREENODE *p, char n[])
{
    p->right = CreateNode(n);
}

```

```

void PrintTree(TREENODE *r,int numspace)
{
    int i;
    if (r)
    {
        for (i=0; i<numspace; i++) printf(" ");
        printf("%s\n",r->name);
        PrintTree(r->left,numspace+4);
        PrintTree(r->right,numspace+4);
    }
}

void WriteFileDetail(TREENODE *r,FILE *f)
{
    if (r)
    {
        fprintf(f,"%c%c%s\n", (r->left)?'+':'-',r->right)?'+':'-',r->name);
        WriteFileDetail(r->left,f);
        WriteFileDetail(r->right,f);
    }
}

void WriteFile(TREENODE *p)
{
    FILE *outfile;
    outfile = fopen("myfile.txt","wt");
    if (!outfile){ printf("Error opening file\nExitig...");exit(0); }
    WriteFileDetail(p,outfile);
    fclose(outfile);
}

void ReadFileDetail(TREENODE *r, FILE *f)
{
    char line[200];
    fgets(line,sizeof(line),f);
    line[strlen(line)-1]='\0'; // eat newline
    strcpy(r->name,line+2);
    if (line[0]=='+')
    {
        r->left = CreateNode("");
        ReadFileDetail(r->left,f);
    }

    if (line[1]=='+')
    {
        r->right = CreateNode("");
        ReadFileDetail(r->right,f);
    }
}

void ReadFile(TREENODE **p)
{
    FILE *infile;
    infile = fopen("myfile.txt","rt");
    if (!infile) { printf("Error opening file\nExitig...");exit(0); }
    *p=CreateNode("");
    ReadFileDetail(*p,infile);
    fclose(infile);
}

```