Lab6

Sheharyar Alam Khan

May 10, 2019

Contents

1 binaryTree.cpp

```
 binaryTree.cpp
****************************************
****************************************
This file contains definitions for the functions declared in binaryTree.hpp
#include <algorithm>
#include "binaryTree.hpp"
This is a default constructor that sets the root equal to nullptr.
BinaryTree::BinaryTree()
{
        root = nullptr;
}
This is a constructor that creates a new root node and sets its data equal to the root_data
    passed as parameter.
BinaryTree::BinaryTree(string root_data)
{
        root = new Node;
        root->data = root_data;
        root->left = nullptr;
        root->right = nullptr;
}
This is a constructor that creates a new root node and sets its data equal to the root_data
    passed as parameter as well as setting the new nodes left and right nodes to the left
    and right nodes passed as parameters.
BinaryTree::BinaryTree(string root_data, BinaryTree left, BinaryTree right)
{
        root = new Node;
        root->data = root_data;
        root->left = left.root;
        root->right = right.root;
}
```

```
this function returns the height of the binary tree from a node down using recursion and
    max from the standard library.
int BinaryTree::height(const Node* n) const
{
        if (n == nullptr) { return 0; }
        else { return 1 + max(height(n->left), height(n->right)); }
}
this function returns the height of the binary tree from the root using recursion and max
    from the standard library.
int BinaryTree::height() const
{
        return height(root);
}
this function checks if the tree is empty.
bool BinaryTree::empty() const
{
        return root == nullptr;
}
this function returns the data of the root.
string BinaryTree::data()
{
        return root->data;
}
this function returns a binaryTree with the left node as a the root.
BinaryTree BinaryTree::left()
{
        BinaryTree result;
        result.root = root->left;
        return result;
}
this function returns a binaryTree with the right node as a the root.
BinaryTree BinaryTree::right()
```

```
{
        BinaryTree result;
        result.root = root->right;
        return result;
}
```

```
****************************************
****************************************
```

2 binaryTree.hpp

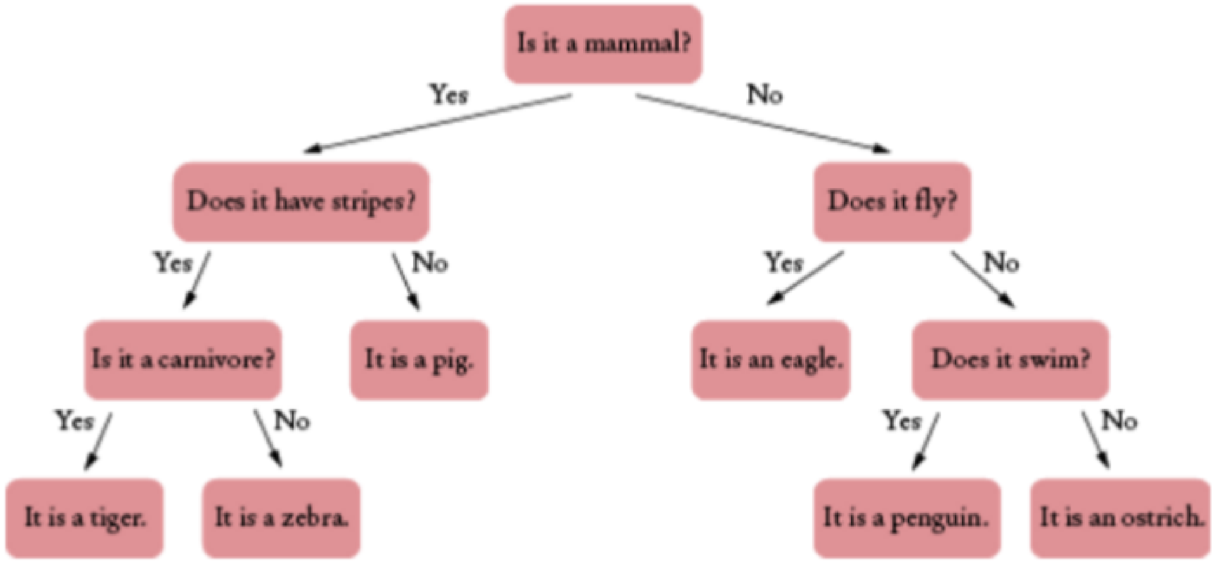
```
 binaryTree.hpp

#ifndef BINARY_TREE_H
#define BINARY_TREE_H
#include <string>

using namespace std;

I have made the Node class public because it is easier to work directly with the Node
    object than with getters and setters in the binary tree
class Node
{
        public:
                string data;
                Node* left = nullptr;
                Node* right = nullptr;
                friend class BinaryTree;
                Node(){}
                Node(string data){this->data=data;}

};

A binary tree in which each node has two children.
class BinaryTree
{

*********************************************************

        public:
```

```
*Constructs an empty tree.
BinaryTree();

*******************************************************

*Constructs a tree with one node and no children.
BinaryTree(string root_data);    *@param root_data the data for the
    root

*******************************************************

*Constructs a binary tree.
BinaryTree(string root_data, BinaryTree left, BinaryTree
    right);
         *@param root_data the data for the root@param left the left
             subtree@param right the right subtree

*******************************************************

*Returns the height of this tree.
int height() const; *returns the height

*******************************************************

*Checks whether this tree is empty.
bool empty() const;      *@return true if this tree is empty

*******************************************************

*Gets the data at the root of this tree.
string data();   *@return the root data
```

```
                ******************************************************

                *Gets the left subtree of this tree.
                BinaryTree left();        *@return the left child of the root

                ******************************************************

                *Gets the right subtree of this tree.
                BinaryTree right();       *@return the right child of the root

******************************************************

                *Returns the height of the subtree whose root is the given node.
                int height(const Node* n) const;
                          *@param n a node or nullptr@return the height of the subtree,
                              or 0 if n is nullptr

                ******************************************************

                This constructor creates a binary tree object with the passed node
                BinaryTree(Node * node){
                          this->root = node;
                }

                Node* root = nullptr;
};
#endif

******************************************
******************************************
```

3 buildTree.cpp

```
buildTree.cpp

#include "binaryTree.hpp"
#include "parseFile.hpp"

This function returns a binaryTree with hardcodes values
BinaryTree buildTree()
{
        BinaryTree question_tree(
                BinaryTree("Is it a mammal?",
                        BinaryTree("Does it have stripes?",
                                BinaryTree("Is it a carnivore?",
                                        BinaryTree("It is a tiger."),
                                        BinaryTree("It is a zebra.")),
                                BinaryTree("It is a pig.")),
                        BinaryTree("Does it fly?",
                                BinaryTree("It is an eagle."),
                                BinaryTree("Does it swim?",
                                        BinaryTree("It is a penguin."),
                                        BinaryTree("It is an ostrich.")))));
        return question_tree;
}
The resulting tree would look like this.
```



This function builds a tree by reading it from a file.

```
BinaryTree buildTree(std::string filename)
{
        BinaryTree tree = readFile(filename);
        return tree;
}
```

4 getImage.cpp

```
 getImage.cpp
****************************************
****************************************

#include <stdio.h>
#include <curl/curl.h>
#include <iostream>
#include <fstream>
#include "mashap.hpp"

 callback function to receive the image
size_t handleData2(char* p, size_t s, size_t n, std::ofstream* u)
{
    size_t ns = s * n;

    u->write(p, ns);

    return s * n; //indicates if there is more data or not
    //zero means no more packets to receive
}

//"https://media3.giphy.com/media/wPud2z0g029Xy/200.gif"

//expect s is a valid URL of a gif and we want to save it in a file
This function uses curl to get an image and save it to the animal.gif file..
void getImage(std::string k)
{
    CURL *curl;

    CURLcode respcode = CURLE_OK;
    std::ofstream ofs("animal.gif", std::ios::binary);

//std::string r = url + api;

curl_global_init(CURL_GLOBAL_DEFAULT);

curl = curl_easy_init();

if(curl)
{
    curl_easy_setopt(curl,
                    CURLOPT_URL, k.c_str());

            //std::cout << r << std::endl;

            curl_easy_setopt(curl,
                    CURLOPT_WRITEFUNCTION, handleData2);

    curl_easy_setopt(curl,
                    CURLOPT_WRITEDATA, &ofs);//change to write to a
                        file

            Perform the request, res will get the return code
    respcode = curl_easy_perform(curl);

            Check for errors
    if(respcode != CURLE_OK)
    {
        fprintf(stderr, "curl_easy_perform() failed: %s\n",
                                curl_easy_strerror(respcode));
    }

              always cleanup
    curl_easy_cleanup(curl);

  }

  curl_global_cleanup();

  //return;
}
```

****
****

5 getImageURL.cpp

```cpp
#include <stdio.h>
#include <curl/curl.h>
#include <iostream>
#include "mashap.hpp"
```

callback function to receive the image

```cpp
size_t handleData(char* p, size_t s, size_t n, std::string* u)
{
    *u += p;

    return s * n; //indicates if there is more data or not
    //zero means no more packets to receive
}


//"https://media3.giphy.com/media/wPud2z0g029Xy/200.gif"
```

This function finds a gif image with a matching keyword k. and returns the image as a JSON object in string format.

```cpp
std::string getImageURL(std::string k)
{
    CURL *curl;
    CURLcode res;
    std::string s = "gif: ";

    //~ struct curl_slist * slist1 = NULL;
    //~ slist1 = curl_slist_append(slist1, key.c_str());
    //~ slist1 = curl_slist_append(slist1, js.c_str());
```

This part shortens the string to the last word by iterating throught string from the end

```
    until it reaches a space.  or the start of the string.
     int start = k.length();
     while(k[start] != ' ' && start >= 0) start--;
     k = k.substr(start + 1, k.length() - 2);
the "&limit=1" limits the search results to 1 json object
     std::string r = url + api + "&q=" + k + "&limit=1";

     curl_global_init(CURL_GLOBAL_DEFAULT);

     curl = curl_easy_init();

     if(curl)
     {
         curl_easy_setopt(curl,
                          CURLOPT_URL, r.c_str());

                 //~ curl\_easy\_setopt(curl,
                          //~ CURLOPT_HTTPHEADER, slist1);

                 std::cout << r << std::endl;

         curl_easy_setopt(curl,
                          CURLOPT_WRITEFUNCTION, handleData);

         curl_easy_setopt(curl,
                          CURLOPT_WRITEDATA, &s);

 Perform the request, res will get the return code
         res = curl_easy_perform(curl);

 Check for errors
         if(res != CURLE_OK)
```

```
        {
            fprintf(stderr, "curl_easy_perform() failed: %s\n",
                    curl_easy_strerror(res));
        }

 always cleanup
        curl_easy_cleanup(curl);
    }

    curl_global_cleanup();

    return s;
}

*****************************************
*****************************************
```

6 getResponse.cpp

```
 getResponse.cpp
****************************************
****************************************

#include <iostream>
#include <string>
#include "binaryTree.hpp"

this fuction gets the users response.  Its for a CLI not fltk.
std::string getResponse(BinaryTree t)
{
        std::string r;
        do
        {
                cout << t.data() << " (y/n): ";
                cin >> r;
        } while (r != "y" && r != "n");
        return r;
}

****************************************
****************************************
```

7 jsonMain.cpp

```
This is the file I used to test my parseJson and string functions.

#include "mashap.hpp"
#include "string.hpp"
#include <iostream>

std::string getImageURL(std::string k);
void getImage(std::string k);

using namespace std;

int main(void)
{
        string query = "cat";
        string json = getImageURL(query);
        std::string originalUrl = "\"original\":{\"url\":\"";
        int start = find(json, originalUrl, 0);
        int end = find(json, ".gif", start + 1);
        int length = end - start + 4 - originalUrl.length();
        string parsedUrl = substr(json, start + originalUrl.length(), length
            );
        string cleanedUrl = clean(parsedUrl, '\\');

        cout << "start, end, length: " << start << ", " << end <<", " <<
            length <<endl;
        cout << "Query: " << query << endl;
        cout << "parsedUrl: " << parsedUrl << endl;
        cout << "cleanedUrl: " << cleanedUrl << endl;
        cout << json;
}
```

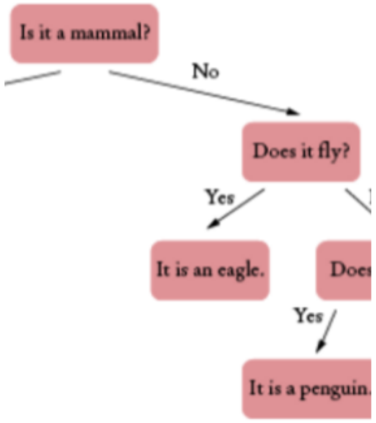
8 lab.cpp

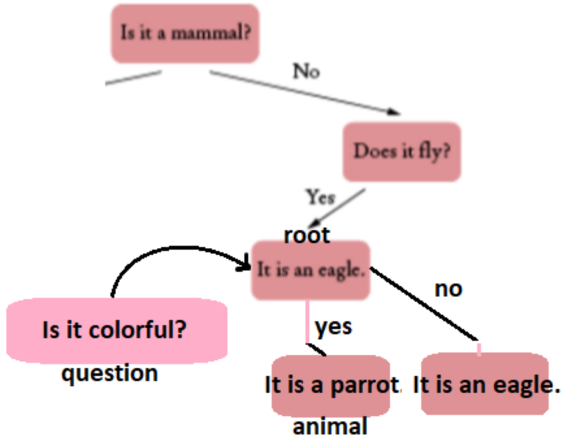
lab.cpp

```
//g++ -g -I/home/debian/fltk-1.3.4-2 -I/home/debian -o "lab" *.cpp -lcurl /
    home/debian/catch2.o `fltk-config --cxxflags --ldflags --use-images --
    use-cairo`
//g++ -g -I/home/debian/fltk-1.3.4-2 -I/home/debian -o "lab" *.cpp -lcurl /
    home/debian/catch2.o `fltk-config --cxxflags --ldflags --use-images --
    use-cairo`


#include <cstdlib>
#include <iostream>
#include <fstream>
#include <config.h>
#include <curl/curl.h>
#include <FL/Fl_Cairo_Window.H>
#include <FL/Fl_GIF_Image.H>
#include <FL/Fl_Anim_GIF_Image.H>
#include <FL/Fl_Box.H>
#include <FL/fl_ask.H>
#include "binaryTree.hpp"
#include "parseFile.hpp"


//Compilation command:
//g++ -g -I/home/debian -o "lab" "lab.cpp" -lcurl getImageURL.cpp -I/home/
    debian/fltk-1.3.4-2 `fltk-config --cxxflags --ldflags --use-images --use
    -cairo`
//g++ -g -I/home/debian -o "lab" "lab.cpp" -lcurl getImage.cpp -I/home/
    debian/fltk-1.3.4-2 `fltk-config --cxxflags --ldflags --use-images --use
```

```
   -cairo`

//Fl_Cairo_Window cw;
const int WIDTH = 600;
const int HEIGHT = 600;
BinaryTree buildTree();
BinaryTree buildTree(std::string filename);
std::string parseJson(std::string j);
std::string getImageURL(std::string);
void getImage(std::string);

int main()
{
        First I build a tree using the build tree function by fassing in the file path.
            The I make a Traveler Node that iterates throught the list and is used to
            access the nodes to ask questions.
        BinaryTree tree = buildTree("animals.txt");
        Node *trav = tree.root;
        Fl_Cairo_Window cw(WIDTH, HEIGHT);

        Fl_Box b(10, 10, WIDTH, HEIGHT);

        string s = "cat"; //default set to cat

        int choice;
        string animal;
        while(true)
        {
                if its a nullptr or the list is empty return 0.  This prevents segfault
                    on empty list
                if(trav == nullptr) return 0;
                if(trav->left == nullptr && trav->right == nullptr)
```

```
{
        when it reaches a leaf node it asks is it guessed the right
            animal.  if it doesnt is asks the user for the correct
            animal and a question that would help it distinguish
            between the corect answer and the wrong answer.
        string q = trav->data+" Is my assessment correct?";
        choice = fl_choice(q.c_str(), "No", "Yes", 0);
        s = trav->data;
        if the user chooses no the program asks the user what his
            animal is and a question that could be used to
            differentiate between the guessed animal and the users
            animal.  it then modifies the tree as follows:
            sets the current node's left child (yes answer) to new
            node with the data animal.
            sets the current node's right child (no answer) to the
            current nodes data.
            replaces the current node's data with the question string.

        if(choice == 0)
        {
                animal = fl_input("Enter new animal: ");
                string question = fl_input("What question
                    would you have asked?");
                animal += ".";
                trav->left = new Node(animal);
                trav->right = new Node(trav->data);
                trav->data = question;
                s = animal;
        }
        Visual representation of the code above is explained in the
            figure below.
```

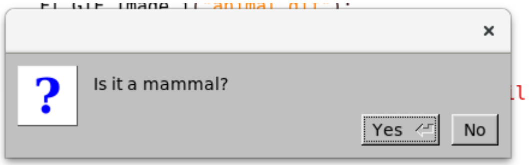


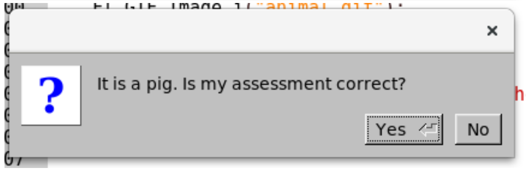


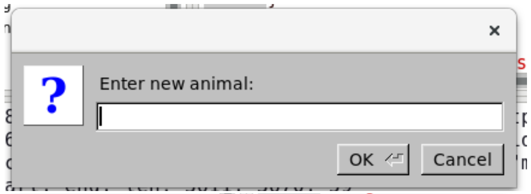
```
                break;
            }
            else
            {
                Over here the program iterates over the list with user input
                choice = fl_choice(trav->data.c_str(), "No", "Yes",
                    0);
                if(choice == 1)
                        trav = trav->left;
                else if(choice == 0)
                        trav = trav->right;
                else
                        break;
            }
    }
    First I get a JSON object in string format using the getImageUrl function passing
```

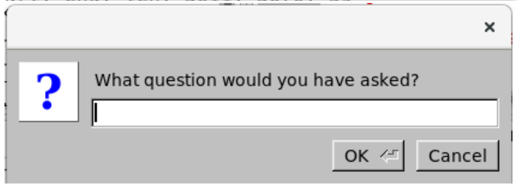
```
            s as a parameter.  Then I parse the json object and updat s.  finally I get
            the image using s which is now a proper url.
        s = getImageURL(s);
        s = parseJson(s);
        getImage(s);
        Fl_GIF_Image i("animal.gif");
        b.image(&i);

        cw.show();
        at the end of the function I update the file by writing the tree to it.
        writeTree(tree ,"animals.txt");
        return Fl::run();

}
```

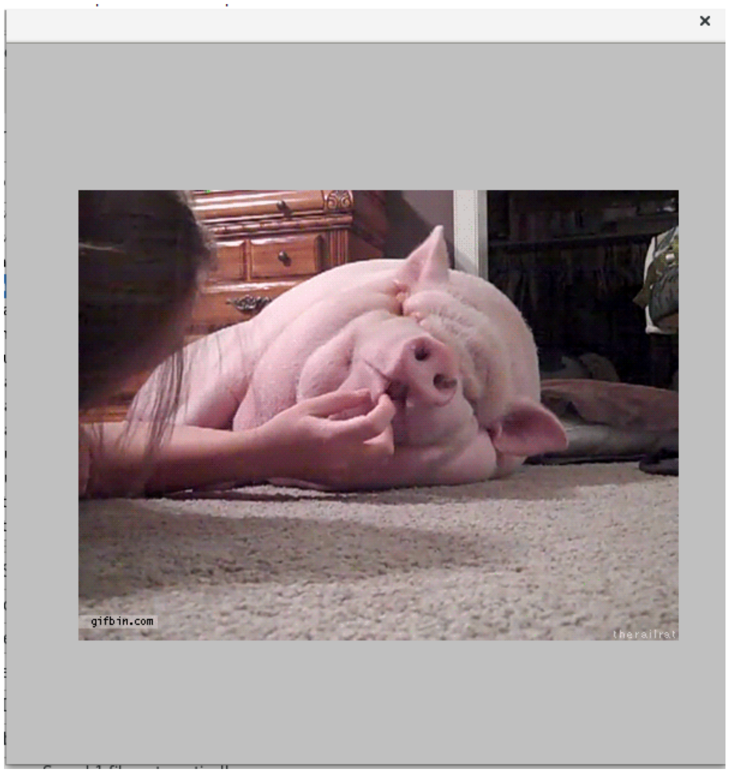
The gui looks as follows:


Is it a mammal?


It is a pig. Is my assessment correct?


Enter new animal:


What question would you have asked?


gifbin.com
therailrat

9 mashap.hpp

```
******************************************************************************************************g++
    -g -I/home/debian -o "lab" "lab.cpp" -lcurl getImage.cpp -I/home/debian/fltk-1.3.4-2
    `fltk-config --cxxflags --ldflags --use-images
    --use-cairo`****************************************************************************************
*****************************************************************************************************
*****************************************************************************************************
 mashape.hpp

this file contains all the constants used in the other files.
*****************************************
*****************************************
#ifndef MASHAP_HPP
#define MASHAP_HPP

#include <string>

const std::string url = "https://api.giphy.com";
const std::string key =
"X-RapidAPI-Key: DrFjbSm0JnmshTt0NowBkUY1WpcXp1bDqRvjsn4MrMqadGpwlM";
const std::string js = "Accept: application/json";
const std::string api = "/v1/gifs/search?api_key=
    evJNRw4gT9hrC9P2hfQPe22czt629zPa";

*****************************************
*****************************************

#endif
```

10 parseFile.cpp

```
#include "parseFile.hpp"

this function writes the data of the node and its path to the file on a single line.  It
    then calls itself by passing in the child nodes.  It writes to the file in preorder.
void writeNode(BinaryTree tree, ofstream * file, string path)
{
        if(tree.empty())
                return;
        *file << tree.data() << "(" << path <<")" <<endl;
        writeNode(tree.left(), file, path + "y");
        writeNode(tree.right(), file, path + "n");
}
opens file and calls the writeNode function by passing in the output file and the root of
    the binary tree.
void writeTree(BinaryTree tree, string filename)
{
        ofstream *ofs = new ofstream(filename);
        writeNode(tree, ofs, "");
        ofs->close();
}
This function parses line and creates a node with parsed data.  and then adds child nodes.
    It use a '.'  to determin whether it has reached a leaf node.
void addNode(Node *&root, ifstream *file)
{
        string line, path, data;
        int start, end, len;
        if(!getline(*file, line)) return;
        start = line.find("(") + 1;
        end = line.find(")");
        len = start - end;
        path = line.substr(end-start);
        data = line.substr(0, start - 1);
        root = new Node(data);
        if(data[data.length() - 1] == '.')
                        return;
        addNode(root->left, file);
        addNode(root->right, file);
}
```

This function opens the file. and calls the addNode function by passing in a root node.

```
BinaryTree readFile(string filename){
        Node *root;
        ifstream *file = new ifstream(filename);
        addNode(root, file);
        file->close();
        BinaryTree tree(root);
        return tree;
}
```

11 parseFile.hpp

```
parseFile.hpp

This file contains all the headers for the functions used in parseFile.cpp

#ifndef PARSEFILE_H
#define PARSEFILE_H

#include "binaryTree.hpp"
#include <string>
#include <fstream>
#include <sstream>
#include <iostream>

void writeNode(BinaryTree tree, ofstream * file);
void writeTree(BinaryTree tree, string filename);
void addNode(BinaryTree &tree, ifstream *file);
BinaryTree readFile(string filename);

#endif
```

12 parseJson.cpp

```
#include <iostream>
#include "string.hpp"

using namespace std;

This function parses a JSON object and returns the url for an animated gif image.
The JSON Object looks like this:

gif: {"data":[{"type":"gif","id":"EcCIaUBoC55uM","slug":"jtvedit-jtv-alba-villanueva-EcCIaUBoC55uM",
"url":"https:\/\/giphy.com\/gifs\/jtvedit-jtv-alba-villanueva-EcCIaUBoC55uM","bitly_gif_url":"https:
\/\/gph.is\/1XYMZ6z","bitly_url":"https:\/\/gph.is\/1XYMZ6z","embed_url":"https:\/\/giphy.com\/embed
\/EcCIaUBoC55uM","username":"","source":"https:\/\/janethevirgin-gifs.tumblr.com\/post\/135338633079
\/when-you-are-upset-you-need-to-think-of-things","rating":"g","content_url":"",
"source_tld":"janethevirgin-gifs.tumblr.com",
"source_post_url":"https:\/\/janethevirgin-gifs.tumblr.com\/post\/135338633079\/when-you-are-upset-you-need-to-t
"import_datetime":"2016-03-01 21:55:38","trending_datetime":"0000-00-00 00:00:00",
"images":{"fixed_height_still":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200_s.gif",
"width":"338","height":"200"},"original_still":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy_s
"width":"245","height":"145"},"fixed_width":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200w.gif",
"width":"200","height":"118","size":"524835","mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200w.mp4",
"mp4_size":"34024","webp":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200w.webp","webp_size":"240042"},
"fixed_height_small_still":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/100_s.gif","width":"169",
"height":"100"},"fixed_height_downsampled":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200_d.gif",
"width":"338","height":"200","size":"255740","webp":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200_d.web
"webp_size":"95632"},"preview":{"width":"244","height":"144",
"mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy-preview.mp4","mp4_size":"31442"},
"fixed_height_small":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/100.gif","width":"169",
"height":"100","size":"380867","mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/100.mp4","mp4_size":"257
"webp":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/100.webp","webp_size":"182046"},
"downsized_still":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy-downsized_s.gif","width":"245"
```

```
"height":"145","size":"27571"},"downsized":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy-downs
"width":"245","height":"145","size":"829415"},"downsized_large":{"url":"https:\/\/media3.giphy.com\/media\/EcCIa
"fixed_width_small_still":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/100w_s.gif","width":"100","h
"preview_webp":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy-preview.webp","width":"161",
"height":"95","size":"48712"},"fixed_width_still":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200w
"width":"200","height":"118"},"fixed_width_small":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/100w
"width":"100","height":"59","size":"140688","mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/100w.mp4",
"mp4_size":"13348","webp":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/100w.webp","webp_size":"79202"},
"downsized_small":{"width":"244","height":"144","mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy-d
"mp4_size":"57982"},"fixed_width_downsampled":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200w_d.g
"height":"118","size":"99407","webp":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200w_d.webp","webp_size"
"downsized_medium":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy.gif","width":"245","height":"
"original":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy.gif","width":"245","height":"145","si
"mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy.mp4","mp4_size":"149617","webp":"https:\/\/media3
"webp_size":"379300"},"fixed_height":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200.gif","width":
"mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/200.mp4","mp4_size":"70963","webp":"https:\/\/media3.gi
"webp_size":"553938"},"looping":{"mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy-loop.mp4","mp4_s
"original_mp4":{"width":"480","height":"284","mp4":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy.mp4"
"preview_gif":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy-preview.gif","width":"118","height
"480w_still":{"url":"https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/480w_s.jpg","width":"480","height":"284"}
"title":"ivonne coll c GIF","analytics":{"onload":{"url":"https:\/\/giphy-analytics.giphy.com\/simple_analytics?
"onsent":{"url":"https:\/\/giphy-analytics.giphy.com\/simple_analytics?response_id=5cd61b37704243556b5dd5f3&even
"pagination":{"total_count":38011,"count":1,"offset":0},"meta":{"status":200,"msg":"OK","response_id":"5cd61b377
```

as you can see there's more than one "original" in the string. so we can't just search for original. we must search for the original we want so we must add a few extra characters after original to find the original we want. The image we want is the "original" but the json contains "original_still" and "original_mp4" so we add a couple of characters. In order to make it easy for us we add characters up to right before the url begins. so the string we search for is '"original":"url":"' and add the number of characters in the string we searched for so that start equals the index right before "http" in the string. we want to search till the end of the url. the url ends with ".gif" so we search from the start till we find ".gif" but since the find function will

return the index at which ".gif" starts we must add the length of ".gif" which is 4 characters. Next we get the url using the sub string function but it contains alot of unwanted "
" characters so we use the clean function to clean it. The transformation of the string can be seen below.

```
start, end, length: 3726, 3800, 59
Query: cat
parsedUrl: https:\/\/media3.giphy.com\/media\/EcCIaUBoC55uM\/giphy.gif"
cleanedUrl: https://media3.giphy.com/media/EcCIaUBoC55uM/giphy.gif"
```

```
string parseJson(std::string j)
{
        cout <<"input: " << j <<endl;
        string original = "\"original\":{\"url\":\"";
        int start = find(j, original, 0) + original.length();
        int end = find(j, ".gif", start + 1) + 4;
        int len = end - start;
        cout << "start, end, len: " << start<<", "<< end <<", "<< len <<endl
            ;
        j = substr(j, start, len);
        cout <<"substr: " << j <<endl;
        j = clean(j, '\\');
        cout <<"clean: " << j <<endl;
        return j;

}
```

13 string.cpp

```
This file contains the string functions I used.  Although it was not required I wanted to
    show you that I am more than capable of creating a simple string parsing function.  I
    doubt any other student wrote their own find, substr, and clean functions so I can
    assure you my code is unique.

#include "string.hpp"
#include <iostream>

using namespace std;

Instead of using the find function available to us in the string class I have made my own
    find function that take in 3 parameters.  str is the string being searched.  substr is
    the string being searched through.  start is starting index from which the search
    begins.

int find(string str, string substr, int start)
{
        int i, j;

        j = start;

        while(j < str.length())
        {
                i = 0; //reset i to 0
                while (str[j + i] == substr[i] && // while the characters
                    are the same
                                i < substr.length() &&  // while substr[i]
                                    is still a character
                                i + j < str.length())   // while str[i + j]
                                    is still a character
                    i++;
                if(i == substr.length()) // if all the characters of the
                    substr have been found
                    return j; // return the index of the first character
                j++;//increment j
        }
        return -1;// if substr not found return -1 as error
}
```

Instead of using the substr function found in the string class I have made my own substr function. It takes in 3 parameters. str is the string from which we will create a substring. start is the starting index of the substring. length is the number of character after start that need to be copied to the substring. if the starting index is greater than the length of the input string then we return a string containing a null character and exit the function. Checking this early on prevetns segmentation faults. Next we create an empty string sub and initialize an integer i to 0. then we begin copying over the string by concatonating character by charater until the number characters to be copied over is reached or the end of the input string is reached. finally we return the substring.

```
string substr(string str, int start, int length)
{
        if(start > str.length())
                return "\0";
        string sub = "";
        int i = 0;
        while(start < str.length() //while end of string is not reached
                                && i < length)// while there are still
                                    characters to be copied over.
        {
                sub += str[start];
                i++;
                start++;
        }
        return sub;
}
```

The clean function removes the character passed as parameter from the string passed as parameter and returns a cleaned string s. It does this by creating an empty string and iterating through the input string while concatonating each character that is not the character passed in as parameter to the cleaned string. after it has iterated through of the characters in the string it returns the cleaned string.

```
string clean(string str, char c)
{
        string s = "";
        int i = 0;
        while(i < str.length())
        {
                if(str[i] != c)
                        s += str[i];
                i++;
        }
        return s;
}
```

14 string.hpp

This file contains all the function declarations for the string.cpp file.

```
#ifndef STRING_HPP
#define STRING_HPP

#include <string>
using namespace std;

int find(string str, string substr, int start);
string substr(string str, int start, int length);
string clean(string str, char c);

#endif
```

15 treeDemo.cpp

```
 treeDemo.cpp
 This program demonstrates a decision tree for an animal guessing game.

#include <fstream>
#include <iostream>
#include <string>
#include "binaryTree.hpp"

using namespace std;

BinaryTree buildTree();
std::string getResponse(BinaryTree);


void preOrder(std::ofstream& o, BinaryTree bt)
{
        if (bt.empty())
        {
                return;
        }

        o << bt.data() << std::endl;

        preOrder(o, bt.left());

        preOrder(o, bt.right());

}
```

```
int main()
{
         Build a binary tree called question tree
        BinaryTree question_tree = buildTree(); //calls
         includegraphics[scale=0.7]questionTree.png

        std::ofstream ofs("animals");

        preOrder(ofs, question_tree);

        //preOrder(std::cout, question_tree);

        bool done = false;

        while (!done)
        {
                BinaryTree left = question_tree.left();
                BinaryTree right = question_tree.right();

                if (left.empty() && right.empty())
                {
                        cout << question_tree.data() << endl;
                        done = true;
                }

                else
                {
                        string response = getResponse(question_tree);

                        if (response == "y"){question_tree = left;}
                        else {question_tree = right;}
                }
        }

}

*****************************************************************************************************************
*****************************************************************************************************************
*****************************************************************************************************************
```

16 unitTesting.cpp

```
These are all the unit tests I ran to check my code was working.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
unitTesting is a Catch v2.0.1 host application.
Run with -? for options

-------------------------------------------------------------------------------
Add
-------------------------------------------------------------------------------
unitTesting.cpp:10
...............................................................................

unitTesting.cpp:12:
PASSED:
  CHECK( (tree.left()).empty() == true )
with expansion:
  true == true

unitTesting.cpp:14:
PASSED:
  CHECK( (tree.left()).empty() == false )
with expansion:
  false == false

unitTesting.cpp:15:
PASSED:
  CHECK( (tree.left()).data() == "Left" )
with expansion:
  "Left" == "Left"

-------------------------------------------------------------------------------
```

```
Traverse Right
-------------------------------------------------------------------------------
unitTesting.cpp:18
...............................................................................

unitTesting.cpp:22:
PASSED:
  CHECK( trav->data == "root" )
with expansion:
  "root" == "root"

unitTesting.cpp:24:
PASSED:
  CHECK( trav->data == "Right" )
with expansion:
  "Right" == "Right"

-------------------------------------------------------------------------------
Traverse Left
-------------------------------------------------------------------------------
unitTesting.cpp:27
...............................................................................

unitTesting.cpp:29:
PASSED:
  CHECK( tree.data() == "root" )
with expansion:
  "root" == "root"

unitTesting.cpp:31:
PASSED:
  CHECK( tree.data() == "Left" )
```

```
with expansion:
  "Left" == "Left"

===============================================================================
All tests passed (7 assertions in 3 test cases)
```

```
#include "catch.hpp"
#include "binaryTree.hpp"
#include <iostream>
#include <string>

using namespace std;

BinaryTree tree("root");

TEST_CASE("Add")
{
        CHECK((tree.left()).empty() == true);
        tree.root->left = new Node("Left");
        CHECK((tree.left()).empty() == false);
        CHECK((tree.left()).data() == "Left");
}

TEST_CASE("Traverse Right")
{
        tree.root->right = new Node("Right");
        Node* trav = tree.root;
        CHECK(trav->data == "root");
        trav = trav->right;
        CHECK(trav->data == "Right");
}
```

```
TEST_CASE("Traverse Left")
{
        CHECK(tree.data() == "root");
        tree = tree.left();
        CHECK(tree.data() == "Left");
}

//g++ -g -I/home/debian -o "unitTesting" "*.cpp" ~/catch2.o
```