Joseph Baruch

CS121

Professor: Bruce Bolden

Program 5: TV Series Binary Search Tree

Due: 4/26/2023

Program Design

Objective

Test our knowledge of reading files, character arrays and binary search trees. Implement a binary search tree that will hold information for a tv series including name, category, URL, start year, end year and actor/actress names.

Program Description

This program will open and read a file containing tv series information while simultaneous creating a binary search tree sorted according to the starting data of the series. Once the file is completely read and the binary search tree is created, different functions can be called to print out certain information.

Requirements

1. Functions
   1. Print out all show names in the entire tree.
   2. Print the actor names of a given show in the tree. These shows include.:
      1. NCIS
      2. McHale’s Navy
      3. The Prisoner
      4. The Office
      5. F Troop
      6. Mary Tyler Moore
   3. Print the show names of a given actor in the tree. These actors include:
      1. Bill Daily
      2. Dana Elcar
      3. Andy Griffith
      4. Trees MacNeille
      5. Chris Potter
      6. Dick Sargent
   4. Print all the shows that premiered within a given decade. These decades include:
      1. 2005-2015
      2. 1990-2000
2. Testing
   1. Read in all tv show information from an external file that can be found on Bruce Boldens website.
   2. Print all information in an intentional and organized way.

Program Contents

1. List class (private)
   1. Struct Node: Node of the linked list.
      1. String actN: Name of actor that will be added to the linked list.
      2. Node \* next: Next pointer.
   2. Typedef struct Node \* nodePtr: nodePtr variable type for easier declaration later.
   3. nodePtr head: Head pointer.
2. List class (public)
   1. Functions
      1. List(): Constructor that initializes head = NULL.
      2. Void addNode(string a): Adds a node to the linked list. Does this without recursion.
         1. nodePtr t: New nodePtr which will be initialized and placed at the front of the linked list.
      3. Void print(): Print the linked list.
         1. nodePtr p: Auxiliary pointer to help traverse the linked list.
      4. Bool find(string a): Finds if a linked list has a certain string.
         1. nodePtr p: Auxiliary pointer to help traverse the linked list.
3. Tree Class (private)
   1. Struct BSTreeNode: Node for the Binary search tree.
      1. List actorList: Linked list for actor names.
      2. char line[MAX\_LINE]: Temporary character array to help initialize the rest of the information.
      3. char seriesName[MAX\_LINE]: Name of the series.
      4. char seriesCategory[MAX\_LINE]: Name of the Category of the series.
      5. char seriesURL[MAX\_LINE]: URL of the series.
      6. int yStart: Starting year of the series.
      7. int yEnd: Ending year of the series.
      8. BSTreeNode \* leftPtr: Left pointer of the binary search tree.
      9. BSTreeNode \* rightPtr: Right pointer of the binary search tree.
4. Tree Class (public)
   1. Functions
      1. Tree(): Constructor that initializes the rootPtr = NULL.
      2. Void AddSeries( TreePtr& t, char name[], char cate[], char url[], list h, int start, int end): Adds a node to the tree by passing in all of the information for the node.
      3. Void printSeriesName(TreePtr& t): Prints all of the names of the series (recursively).
      4. Void printActorsOf(TreePTr& t, string name): Prints the linked list of actor of a given series ( recursively ). Print the list by calling a print function declared in the list class.
      5. Void printShowsOf(TreePtr&, string act): Print the shows of a given actor (recursively). Evaluates if the list of actors (in each BSTreeNode) constains the actor name or not by calling a find function declared in the list class.
      6. Void printShowsInDecade(TreePtr& t , int start, int end): Print shows of a given decade (recursively).
   2. Typedef BSTreeNode \* TreePtr: TreePtr variable type for easier declaration later in code.
   3. TreePtr rootPtr: RootPtr for the binary search tree.
5. Tree Class Global Variables
   1. Const int MAX\_LINE: Max size of all character arrays (unless declared with the division of two).
6. Main Functions
   1. Int ReadTVFile(): Discussed in Program Operation Description. Reads file, initializes, and declares variables and calls the function that adds a node to the tree.
      1. fIn: File identifier variable.
      2. const in MAX\_LINE: Max size of all character arrays (unless declared with the division of two).
      3. char line[MAX\_LINE]: Temporary character array to help initialize the rest of the information.
      4. char seriesName[MAX\_LINE]: Name of the series.
      5. char seriesCategory[MAX\_LINE]: Name of the series category.
      6. char seriesURL[MAX\_LINE]: Series URL.
      7. char actorName[MAX\_LINE]: Name of the current actor being grabbed front the text file.
      8. int yStart: Start year of the series.
      9. int yEnd: End year of the series
      10. int nSeries: Number of series in text file.
      11. list \* h: Declares new instance of the list class.
   2. Void GetSeriesName(): Uses getline to grab the name of the series.
      1. Int yearStart: Index of when the year starts within the line temporary string.
   3. Void GetYears(char line[], int & yStart, int & yEnd): Get the start and end year of the series. Uses atoi() which turns the char array into an integer.
      1. Char tmpString[8]: Temporary string which holds the years.
      2. Char yearString[16]: Temporary string which holds everything in the string after series name.
      3. Int yearStart: index of the start year.
      4. Int yearEnd: Index of the end year.
   4. Void RemoveBlanksFromString(char s[]): Removes all of the blanks from the passes in string.
      1. Int sLen: Length of the string.
   5. Int IndexOf(char s[], char c): Returns the index of char c in char s[].
      1. Int i: Returned index of the string.
   6. Void GetSubString(char s[], int start, int end, char res[]): Gets a substring of a string according the starting and ending positions.
      1. Int i: Uses in for loop to traverse char arrays.
      2. Int iRes: Uses in for loop to traverse char arrays.
7. Main Variables
   1. Int nSeries: Number of series in the external file.
   2. Char title[]: Title of a series on that will be used to print all the actors of that series.
   3. Char actor[]: Name of an actor that will be used to print all the shows of that actor.
   4. Int printStart: Start year of the decade which will be used to print all shows in that decade.
   5. Int printEnd: End year of the decade which will be used to print all shows in that decade.
8. Main Other
   1. Tree t: Declaration of the tree class.
9. tvDB.txt (External File): File containing series names and all other mentioned information needing to be sorted.

Program Operation Description

* Call a function that reads a text file and initializes the appropriate information from the text file (uses Bruce Boldens readFile.pdf nodes).
  + Opens file and checks to make sure that it is open.
  + Declares all variables.
  + loop (while) until file is completely read.
    - Uses get-line, a find index function, a function that removes blanks from a string to get the following information.
      * seriesName
      * yStart and yEnd
      * seriesCategory
      * seriesUrl
    - loop (while) there are still actors to read.
      * Create a linked list for all the actor names.
    - Call a function which adds a node to the binary search tree. This initializes all information of the node that was just grabbed.
  + Close File
* Call a function that prints all the series names.
  + In-order traversal (recursion) of the tree which prints series name every time it visits a node.
* Call a function that prints all the actors of a given series.
  + In-order traversal (recursion) of the tree which prints the list of actors if the current series matches the name of the series requested (if statement).
* Call a function that prints all the shows of a given actor.
  + In-order traversal (recursion) of the tree which prints the series name if the actor’s name is found in the list of actors of the series. A function within the list class checks this.
* Call a function that prints all the shows that premiered during a given decade.
  + In-order traversal (recursion) of the tree which prints the series name if the start year of the series is between the decade given (if statement).

Programming Log

Total Time:

4/10/2023: 15 min

* I looked over the lecture on how to sort the file. This was in response to Professor Bolden warning that it would be on the future assignment.

4/12/2023: 15 min

* Listened to Professor Bolden lecture on the ‘readFile.pdf’ lecture. This started to give me a better understanding of the assignment.

4/14/2023: 15 min

* I read through the assignment description to get a general understanding of the challenge ahead.
* Won’t be able to start until Sunday or Monday because of trip to Boise.

4/17/2023: 30 min

* Decided to begin this assignment. This quickly led to me not knowing what to do. I don’t understand what the construction or sorting of the tree will look like with so many different types of information. I tried making diagrams and picking the best situation.
* Continued to work on the write up and prepare to ask questions over email or in office hours.

4/18/2023: 2 hours

* Bolden emailed me back and told me to look at some of his CS 112 notes that should help. Still confused but glad to have a clue. Looks like the data will take the form of some sort of array of a struct type.
* Went into office hours and figured out some of the errors that I was getting.
* Started implementing the readFile.pdf information and Bruce Bolden’s Binary Search Tree notes.
* Running into some roadblocks on how to work between reading the files and adding nodes.

4/19/2023: Nothing

* Spent some time outside of this program working on searching and sorting programs.

4/20/2023: 1.5 hour

* Realized after office hours on 4/18 that I need to call the construct-tree function inside the read-file function so that the read-file function can read the whole file.
* After this realization I implemented this into my code and began cleaning everything up. After I finished, my tree construction and read file functions should logically work but I’m sure there are some aspects that I would need to clean up.
* Need to figure out how to evaluate if a letter comes before another in the alphabet (add this later).
* Updated Programming Log and continued the write up.

4/21/2023-4/23/2023: Nothing

4/24/2023: 2.5 hours

* I thought a lot about the program over the weekend and felt confident about going forward. I knew the general idea on what I had to do but also knew there would be some things I didn’t.
* I got almost the entire program done with the four required functions using in-order traversals. I also finished commenting everything.
* I didn’t compile the program out of fear and wanting to do it the next day. I predicted that I would have some issues with the character arrays and knowing if all the actor names had been grabbed.
* I also got caught up in the program write up but was missing most of the details (description of program contents).

4/25/2023: 6 hours

* Fixed some minor bugs with my current code. With little work I got the print series name function and print series in a decade to work great. The problems now are associated with the character arrays for the two other function.
* Realized that within my read file function, I am only reading in one actor name and then overriding it over and over. I need a way to store many actor names. A tutor helped me see a possibility of storing them in the form of an array of string. I implemented this.
* Getting a lot of issues with printing names of actors for a different series which means there is an issue with my string array. Went to office hours to ask about this.
* Learned in Boldens office hours an array of strings is more complicated because the arrays have a non-dynamic size. I learned that whenever I have a non-set size of a list, use a linked list because it is of dynamic size.
* Typed out all the needed code for the list class but was a bit confused on how to implement it. Learned from a tutor that I can declare a new pointer to a list for every time I read in a series. I can than pass the instance of the linked list through a function and initialize the list in my tree node this way.
* Realized in my print shows of a given actor had a copy and paste error. Once I fixed this my program gave the right output and my program was finished.
* Finished the missing comments of the program (list class).
* Updated the program contents with all the information for the list class and added detailed descriptions to all the contents.
* Finished the programming log.
* Ran all the outputs and copied them into this document.
* Copied and pasted in the program.

4/26/2023: N/A

* Turn in assignment at the beginning of class.

Outputs

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

Series Titles:

-NCIS: New Orleans

-Hawaii Five-0

-NCIS: Los Angeles

-American Dad!

-Criminal Minds

-The Office

-Lost

-NCIS

-Star Trek: Enterprise

-Futurama

-3rd Rock from the Sun

-Dexter's Laboratory

-Star Trek: Voyager

-JAG

-ER

-Babylon

-Babylon

-Chicago Hope

-The X-Files

-Star Trek: Deep Space Nine

-Animaniacs

-Walker, Texas Ranger

-Kung Fu: The Legend Continues

-Herman's Head

-Law & Order

-Evening Shade

-Northern Exposure

-Seinfeld

-The Simpsons

-Coach

-Quantum Leap

-Married with Children

-Star Trek: The Next Generation

-Jake and the Fatman

-Matlock

-MacGyver

-MacGyver

-The Cosby Show

-Murder, She Wrote

-Riptide

-Night Court

-Scarecrow and Mrs. King

-The A-Team

-St. Elsewhere

-Family Ties

-Newhart

-The Fall Guy

-Benson

-WKRP in Cincinnati

-Taxi

-Battlestar Galactica

-Mork & Mindy

-CHiPs

-Charlie's Angels

-The Jeffersons

-Wonder Woman

-Happy Days

-Little House on the Prairie

-The Six Million Dollar Man

-Kojak

-The Streets of San Francisco

-The Bob Newhart Show

-Banacek

-M\*A\*S\*H

-All in the Family

-The Odd Couple

-Mary Tyler Moore

-McCloud

-Room

-Mod Squad

-The Prisoner

-The Carol Burnett Show

-Ironside

-Mannix

-The Invaders

-The Flying Nun

-Batman

-Star Trek

-The Big Valley

-Lost in Space

-The Wild Wild West

-Gidget

-F Troop

-Hogan's Heroes

-I Dream of Jeannie

-Voyage to the Bottom of the Sea

-Gomer Pyle: USMC

-The Man from U.N.C.L.E.

-Bewitched

-Gilligan's Island

-Burke's Law

-The Fugitive

-The Lucy Show

-McHale's Navy

-The Beverly Hillbillies

-The Saint

-The Bullwinkle Show

-The Andy Griffith Show

-My Three Sons

-Mr. Lucky

-The Twilight Zone

-Rawhide

-The Many Loves of Dobie Gillis

-Mister Ed

-Leave It to Beaver

-The Honeymooners

-The Phil Silvers Show

-Alfred Hitchcock Presents

-Lassie

-Topper

-Make Room for Daddy

-The Adventures of Ozzie & Harriet

-I Love Lucy

-The Jack Benny Program

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All actors of NCIS:

Joe Spano

Sasha Alexander

Lauren Holly

Rocky Carroll

Brian Dietzen

Cote de Pablo

Sean Murray

Mark Harmon

David McCallum

Pauley Perrette

Michael Weatherly

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All actors of McHale's Navy:

Gavin MacLeod

Bob Hastings

Yoshio Yoda

John Wright

Edson Stroll

Billy Sands

Gary Vinson

Carl Ballantine

Tim Conway

Joe Flynn

Ernest Borgnine

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All actors of The Prisoner:

Leo McKern

Peter Swanwick

Angelo Muscat

Patrick McGoohan

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All actors of The Office:

Mindy Kaling

B.J. Novak

Oscar Nunez

Creed Bratton

Kate Flannery

Phyllis Smith

Angela Kinsey

Brian Baumgartner

Leslie David Baker

Jenna Fischer

John Krasinski

Rainn Wilson

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All actors of F Troop:

Frank DeKova

James Hampton

Melody Patterson

Ken Berry

Larry Storch

Forrest Tucker

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All actors of Mary Tyler Moore:

Cloris Leachman

Betty White

Georgia Engel

Valerie Harper

Ted Knight

Edward Asner

Gavin MacLeod

Mary Tyler Moore

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All shows that star Bill Daily:

The Bob Newhart Show

I Dream of Jeannie

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All shows that star Dana Elcar:

MacGyver

MacGyver

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All shows that star Andy Griffith:

Matlock

The Andy Griffith Show

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All shows that star Tress MacNeille:

Futurama

Animaniacs

The Simpsons

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All shows that star Chris Potter:

Kung Fu: The Legend Continues

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

All shows that star Dick Sargent:

Bewitched

oseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

Shows Priemered between 2005 and 2015:

-NCIS: New Orleans

-Hawaii Five-0

-NCIS: Los Angeles

-American Dad!

-Criminal Minds

-The Office

joseph.baruch@Josephs-Air programvs5 % g++ main.cpp

joseph.baruch@Josephs-Air programvs5 % ./a.out

Shows Priemered between 1990 and 2000:

-Futurama

-3rd Rock from the Sun

-Dexter's Laboratory

-Star Trek: Voyager

-JAG

-ER

-Babylon

-Babylon

-Chicago Hope

-The X-Files

-Star Trek: Deep Space Nine

-Animaniacs

-Walker, Texas Ranger

-Kung Fu: The Legend Continues

-Herman's Head

-Law & Order

-Evening Shade

-Northern Exposure

Program

/\* main.cpp

Joseph Baruch

CS 121

Professor Bolden

Read TV Series File into a Binary Search Tree

Due: Wednesday, April 26, 2023

\*/

#include <iostream>

#include <fstream>

#include <string>

#include <cstdlib>

#include <string.h>

#include <stdlib.h>

#include "tree.cpp"

#include "list.h"

#include "list.cpp"

using namespace std;

// tree class instances declaration

tree t;

// function prototypes

int ReadTVFile();

void GetSeriesName( char line[], char seriesName[] );

void GetYears( char line[], int & yStart, int & yEnd );

int main(){

int nSeries = 0;

nSeries = ReadTVFile();

cout << "\nSeries Titles: " << endl;

t.printSeriesName( t.rootPtr );

cout << endl;

// NCIS, McHale‚Äôs Navy, The Prisoner, The Office, F Troop, Mary Tyler Moore

char title[] = {"Mary Tyler Moore"};

cout << "\nAll actors of " << title << ":" << endl;

t.printActorsOf( t.rootPtr, title );

cout << endl;

// Bill Daily, Dana Elcar, Andy Griffith, Tress MacNeille, Chris Potter, Dick Sargent

char actor[] = {"Dick Sargent"};

cout << "\nAll shows that star " << actor << ":" << endl;

t.printShowsOf( t.rootPtr, actor );

cout << endl;

// (2005, 2015), (1990, 2000)

int printStart = 1990;

int printEnd = 2000;

cout << "\nShows Priemered between " << printStart << " and " << printEnd << ":" << endl;

t.printShowsInDecade( t.rootPtr, printStart, printEnd );

cout << endl;

return EXIT\_SUCCESS;

}

int IndexOf( char s[], char c ){ // finds the index (in the array) of a char

int i = 0;

// while current position in array isnt '\0' or c

while( s[i] != '\0' && s[i] != c )

i++; // increment current position to evaluate in loop

return i;

}

int ReadTVFile(){ // Reads in the TV Series File and calls AddSeries()

ifstream fIn( "tvDB.txt", ios::in ); // opens file

if( !fIn ){ //checks to see if file is open

cout << "Unable to open \"tvDB\" data file" << endl;

exit( -1 ); // end program if file isn't open

}

// ------ Variable Declarations ----------

const int MAX\_LINE = 128;

char line[MAX\_LINE];

char seriesName[MAX\_LINE];

char seriesCategory[MAX\_LINE/2];

char seriesURL[MAX\_LINE];

char actorName[MAX\_LINE/2];

int yStart, yEnd;

int nSeries = 0;

while( fIn.getline( line, MAX\_LINE ) ){ // while file isnt fully read

if( strlen(line) == 0 ) // if length of string ins't empty

continue;

GetSeriesName( line, seriesName ); // get series name

GetYears( line, yStart, yEnd ); // get series years

fIn.getline( seriesCategory, MAX\_LINE/2 ); // getline for series category

fIn.getline( seriesURL, MAX\_LINE ); // getline for series URL

fIn.getline( line, MAX\_LINE/2 ); // getline for actors

list \* h = new list; // declare a new pointer to a list instance

while( strlen(line) > 0 ){ // while line is greater than 0

strcpy( actorName, line ); // copy string for actor name

h->addNode( actorName ); // add actor to front of list

fIn.getline( line, MAX\_LINE/2 ); // new getline if more than one actor

}

nSeries++; // increment the number of series counter

// -------- Add current series to the tree ------------

t.AddSeries( t.rootPtr, seriesName, seriesCategory, seriesURL, \*h, yStart, yEnd );

}

fIn.close(); // close the file

return nSeries;

}

void RemoveBlanksFromString( char s[] ){ // removes blanks from string in GetSubString()

int sLen = strlen( s );

for( int i = sLen ; i >= 0 ; i-- ){ // works back to front in array

if( isalpha(s[i]) )

break;

if( s[i] == ' ' ) // if array index is empty

s[i] = '\0'; // set as '\0'

}

}

void GetSubString( char s[], int start, int end, char res[] ){ // gets Substring to be set to result array

int i;

int iRes = 0; // index of result string initialization

for( i = start ; i < start+end ; i++ ){

res[iRes++] = s[i]; // copies from s[] into res[] at appropriate positions

}

res[iRes] = '\0';

RemoveBlanksFromString( res ); // remove the blanks from resulting string

}

void GetSeriesName( char line[], char seriesName[] ){ // get series name

int yearStart;

yearStart = IndexOf( line, '(' ); // find index of '(' (end of series name)

// gets series name by stopping when the current position in '('

GetSubString( line, 0, yearStart-1, seriesName );

}

void GetYears( char line[], int & yStart, int & yEnd ){ // get yStart and yEnd

char tmpString[8];

char yearString[16];

int yearStart, yearEnd;

yearStart = IndexOf( line, '(' ); // yStart starts right after '('

yearEnd = IndexOf( line, ')' ); // yEnd ends right before ')'

// --------------- Get both years together -------------------

GetSubString( line, yearStart+1, yearEnd-yearStart-1, yearString );

// --------- Seperate out yStart -----------

GetSubString( yearString, 0, 4, tmpString );

yStart = atoi( tmpString ); // set yStart as an integer value

// ------------------- Separate out yEnd --------------------

GetSubString( yearString, 7, 4, tmpString ); // 7? not ‚Äô-‚Äô!

yEnd = atoi( tmpString ); // set yEnd as an integer value

}

/\* tree.h

Joseph Baruch

\*/

#ifndef TREE\_H // file guard

#define TREE\_H

#include <iostream>

#include <string>

#include "list.h"

using namespace std;

// contstant array size

const int MAX\_LINE = 128;

class tree{

private:

// BSTreeNode that constains all required series information

struct BSTreeNode{

list actorList; // instance of linked list

char line[MAX\_LINE];

char seriesName[MAX\_LINE];

char seriesCategory[MAX\_LINE/2];

char seriesURL[MAX\_LINE];

string actorsName[MAX\_LINE];

int yStart, yEnd;

BSTreeNode \* leftPtr; // left pointer

BSTreeNode \* rightPtr; // right pointer

};

public:

typedef BSTreeNode \* TreePtr; // declaration of TreePtr variable type

TreePtr rootPtr; // declare rootPtr ( initialized in tree() )

tree(); // constructor

// ---------------------- Adds series to tree ( called in ReadTVFile ) ---------------------------

void AddSeries( TreePtr& t, char name[], char cate[], char url[], list h, int start, int end );

void printActorsOf(TreePtr& t, string name ); // print all actor names of a given show

void printShowsOf(TreePtr& t, string act ); // print all shows of a given actor

void printShowsInDecade( TreePtr& t, int start, int end ); // print all shows primered in a given decade

void printSeriesName( TreePtr& t ); // print all show names

};

#endif // end file guard

/\* tree.cpp

Joseph Baruch

\*/

#include <iostream>

#include <cstdlib>

#include <string>

#include "tree.h"

using namespace std;

// ------ constructor -----

tree::tree(){

rootPtr = NULL; // intialize the rootPtr = NULL

}

// ----- Add Series information (within BSTreeNode) to tree -------

void tree::AddSeries( TreePtr& t, char name[], char cate[],

char url[], list h, int start, int end ){

if( t == NULL ){ // if at the end of tree, place node

// allocate new TreePtr

TreePtr newPtr = new BSTreeNode;

// initialize all information for series in new TreePtr

for(int i = 0; i < MAX\_LINE; i++ ){

newPtr->seriesName[i] = name[i];

newPtr->seriesCategory[i] = cate[i];

newPtr->seriesURL[i] = url[i];

}

newPtr->actorList = h;

newPtr->yStart = start;

newPtr->yEnd = end;

// set left and right pointers = NULL

newPtr->leftPtr = NULL;

newPtr->rightPtr = NULL;

t = newPtr; // place

// if current position in tree's start year is less than start

}else if( t->yStart <= start ){

// go left in tree

AddSeries( t->leftPtr, name, cate, url, h, start, end );

// if current position in tree's start year is greater than start

}else{

AddSeries( t->rightPtr, name, cate, url, h, start, end );

}

}

// --------- print all Series Names ----------

void tree::printSeriesName( TreePtr& t ){

// printing using in-order traversal (recursion)

if( t != NULL ) {

printSeriesName( t->leftPtr );

cout << "-" << t->seriesName << endl;

printSeriesName( t->rightPtr );

}

}

// ------- print actors of a specific show ---------

void tree::printActorsOf( TreePtr& t, string a ){

// printing using in-order traversal (recursion)

if( t != NULL ) {

printActorsOf( t->leftPtr, a );

// if current series name matches given series name

if( t->seriesName == a ){

// print list of actor names

t->actorList.print();

}

printActorsOf( t->rightPtr, a );

}

}

// ------ print all shows of a given actor -----

void tree::printShowsOf(TreePtr& t, string act ){

// printing using in-order traversal (recursion)

if( t != NULL ) {

printShowsOf( t->leftPtr, act );

// if actor is found for series in list

if( t->actorList.find( act ) ){

cout << t->seriesName << endl;

}

printShowsOf( t->rightPtr, act );

}

}

// ----- print all shows that premiered within a given decade ---

void tree::printShowsInDecade( TreePtr& t, int start, int end ){

// printing using in-order traversal (recursion)

if( t != NULL ) {

printShowsInDecade( t->leftPtr, start, end );

// if start is within given decade

if( start <= t->yStart && end >= t->yStart ){

cout << "-" << t->seriesName << endl;

}

printShowsInDecade( t->rightPtr, start, end );

}

}

/\* link.h

Joseph Baurch\*/

#ifndef LINK\_H

#define LINK\_H

#include <iostream>

using namespace std;

class list{

private:

struct Node{

string actN;

Node \* next;

};

typedef struct Node \* nodePtr;

nodePtr head; // creates head nodePtr

public:

list(); // sets head = NULL

void addNode(string a); // add node to list

void print(); // print list

bool find( string a ); // sees if list contains string

};

#endif

/\* list.cpp

Joseph Baruch

\*/

#include <iostream>

#include "list.h"

using namespace std;

list::list(){

head = NULL; // sets head = null

}

// adds node to the front of the linked list

void list::addNode( string a ){

nodePtr t = new Node;

t->actN = a;

t->next = NULL;

if( head == NULL ){

head = t;

}else{

t->next = head;

head = t;

}

}

// prints linked list

void list::print(){

nodePtr p = head;

while( p != NULL ){

cout << p->actN << endl;

p = p->next;

}

}

// returns true if string a is in list

bool list::find( string a ){

nodePtr p = head; // aux pointer

while( p != NULL ){

if( p->actN == a){

return true;

}

p = p->next;

}

// return false if a not found

return false;

}