CS 251
Data Structures
Spring 2017

Project #02: Netflix Movies and Binary Search Trees

Complete By: Wednesday, January 25th @ 11:59pm

Assignment: C program to build & search binary search trees

Policy: Individual work only, late work *is* accepted (see "Policy" section on last page for more details)

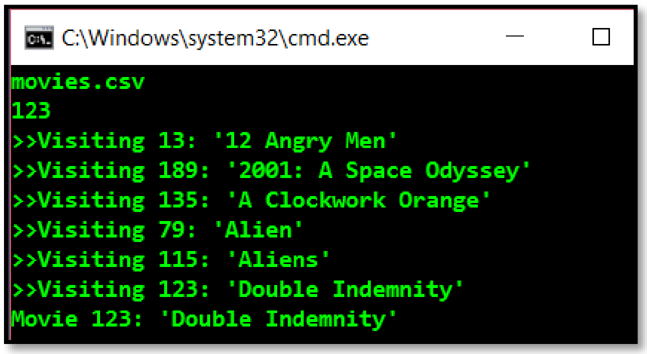
Submission: online via zyLabs ("Project02 BST Netflix Movies", section 4.7 --- you may need to refresh browser)

Assignment

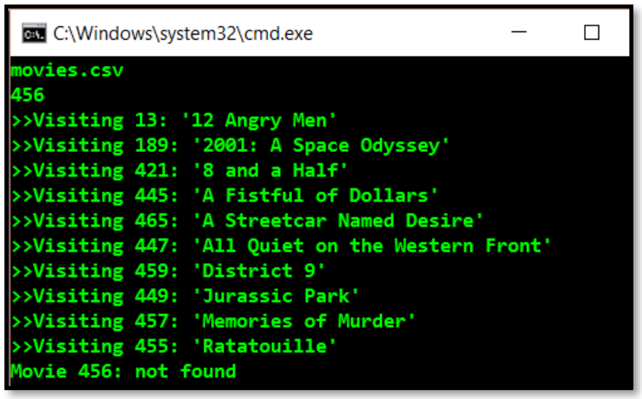
The assignment is to input Netflix movie data, store the data in a binary search tree (using the movie id as the key), and then search for a movie id entered by the user. As the program visits nodes during the search, the movie id and name stored in each node is output. The program then outputs the movie id and name if found in the tree, otherwise a "not found" message is output.

The screenshots to the right depict two runs of the program. In the first run, the user is specifying the program to input from the file "movies.csv", and to search for movie 123. The program outputs the movies that are visited during the search, and then finds the movie and outputs both the id and the name. In the second run, the user inputs the same data, and searches for movie 456, which is not found.

Similar to project #01, the movie data file is in CSV format; more detail is given on the next page. Note that your program must match the screenshots exactly, since it will be graded using the zyLabs system.


C:\Windows\system32\cmd.exe

```
movies.csv
123
>>Visiting 13: '12 Angry Men'
>>Visiting 189: '2001: A Space Odyssey'
>>Visiting 135: 'A Clockwork Orange'
>>Visiting 79: 'Alien'
>>Visiting 115: 'Aliens'
>>Visiting 123: 'Double Indemnity'
Movie 123: 'Double Indemnity'
```


C:\Windows\system32\cmd.exe

```
movies.csv
456
>>Visiting 13: '12 Angry Men'
>>Visiting 189: '2001: A Space Odyssey'
>>Visiting 421: '8 and a Half'
>>Visiting 445: 'A Fistful of Dollars'
>>Visiting 465: 'A Streetcar Named Desire'
>>Visiting 447: 'All Quiet on the Western Front'
>>Visiting 459: 'District 9'
>>Visiting 449: 'Jurassic Park'
>>Visiting 457: 'Memories of Murder'
>>Visiting 455: 'Ratatouille'
Movie 456: not found
```

Input Files

The input file is a text file in **CSV** format — i.e. comma-separated values. The first file defines the set of **movies**, in no particular order (could be sorted by movie name, could be sorted by publication year, or could be entirely random). An example of the format is as follows:

```
MovieID,MovieName,PubYear
13,12 Angry Men,1957
189,2001: A Space Odyssey,1968
421,8 and a Half,1963
  .
  .
  .
257,Wild Strawberries,1957
199,Witness for the Prosecution,1957
213,Yojimbo,1961
```

The first line contains column headers, and should be ignored. The data starts on line 2, and each data line contains 3 values: a movie ID (integer), the movie name (string, at most 255 characters), and the year the movie was published (integer). Note that this is just one possible input file, we will use different files when grading. You may assume N > 0 movies, but make no other assumptions about the data. In particular, note that the movie ids are no longer consecutive, i.e. do not assume 1, 2, 3, 4, ...

At least two input files are available on the course web page under "Projects", then "project02-files". The files are being made available for each platform (Linux, Mac, and PC) so you may work and test on your platform of choice. Even though the files end in the file extension ".csv", these are text files that can be opened in a text editor (e.g. Notepad) or a spreadsheet program (e.g. Excel). **Note**: the best way to download the files is *not* to click on them directly in dropbox, but instead use the "download" button in the upper-right corner of the dropbox web page.

Requirements

In CS 251, how you solve the problem is as important as getting the correct answer. To encourage good programming practices, and use of appropriate data structures, your solution must meet the following requirements:

- You must use a binary search tree to store the data, and to lookup the movie. The design must support (key, value) pairs, a BST handle, and BSTNodes.
- Use BST functions as we have been doing in lab, the homework, and class. This means a minimum of at least the following functions: BSTCreate, BSTCompareKeys, BSTInsert, and BSTSearch.
- You must dynamically malloc your BSTNodes; you are not required to free the memory.
- No global variables whatsoever; use functions and parameter passing.

We are serious about these requirements. No binary search tree. Score of 0. Use global variables instead of passing parameters? Score of 0.

Programming Environment

You are welcome to program on whatever platform you want, using whatever compiler / programming environment you want; common options were presented in the syllabus. To facilitate testing, we'll be using the zyLabs system for submission and grading; see "**Project02 BST Netflix Movies**", section 4.7. You can work entirely within zyLabs in "Develop" mode, or you can download the input files from the course web page (along with a sample "main.c" to get you started) and work outside zyLabs.

When you are ready to submit your program, copy-paste your program if necessary, switch to "Submit" mode, and click SUBMIT FOR GRADING.

Have a question? Use Piazza, not email

As discussed in the syllabus, questions must be posted to our course Piazza site — questions via email will be ignored. Remember the guidelines for using Piazza:

1. *Look before you post* — the main advantage of Piazza is that common questions are already answered, so search for an existing answer before you post a question. Posts are categorized to help you search, e.g. "Pre-class" or "HW".
2. Post publicly — only post privately when asked by the staff, or when it's absolutely necessary (e.g. the question is of a personal nature). Private posts defeat the purpose of piazza, which is answering questions to the benefit of everyone.
3. Ask pointed questions — do not post a big chunk of code and then ask "help, please fix this". Staff and other students are willing to help, but we aren't going to type in that chunk of code to find the error. You need to narrow down the problem, and ask a pointed question, e.g. "on the 3rd line I get this error, I don't understand what that means…".
4. Post a screenshot — sometimes a picture captures the essence of your question better than text. Piazza allows the posting of images, so don't hesitate to take a screenshot and post; see http://www.take-a-screenshot.org/ .
5. Don't post your entire answer — if you do, you just gave away the answer to the ENTIRE CLASS. Sometimes you will need to post code when asking a question --- in that case post only the fragment that denotes your question, and omit whatever details you can. If you must post the entire code, then do so privately --- there's an option to create a private post ("visible to staff only").

Submission

The program is to be submitted via zyLabs: ; see "**Project02 BST Netflix Movies**", section 4.7. The grading scheme at this point is 90% correctness, 10% style — we expect basic commenting, indentation, and readability. You should also be using functions, parameter passing, and good naming conventions.

Policy

Late work *is* accepted. You may submit as late as 24 hours after the deadline for a penalty of 25%. After 24 hours, no submissions will be accepted.

All work is to be done individually — group work is not allowed. While I encourage you to talk to your peers and learn from them (e.g. via Piazza), this interaction must be superficial with regards to all work submitted for grading. This means you *cannot* work in teams, you cannot work side-by-side, you cannot submit someone else's work (partial or complete) as your own. The University's policy is described here:

http://www.uic.edu/depts/dos/docs/Student%20Disciplinary%20Policy.pdf .

In particular, note that you are guilty of academic dishonesty if you extend or receive any kind of unauthorized assistance. Absolutely no transfer of program code between students is permitted (paper or electronic), and you may not solicit code from family, friends, or online forums. Other examples of academic dishonesty include emailing your program to another student, copying-pasting code from the internet, working in a group on a homework assignment, and allowing a tutor, TA, or another individual to write an answer for you. Academic dishonesty is unacceptable, and penalties range from failure to expulsion from the university; cases are handled via the official student conduct process described at http://www.uic.edu/depts/dos/studentconductprocess.shtml .