Data Structures (CS 1520) Homework #4 Due: Oct. 28 (Sat.) at 11:59 PM

Objective: To gain experience using a hash-table based dictionary by implementing a movie search.

To start the project: Download `hw4.zip` file and extract it. The `hw4` directory contains the lecture 15 `ChainingDict` dictionary implementation (in `chaining_dictionary.py`) which you can use.

The Assignment: Professor Bob is a movie buff, but like most professors he is absent-minded. He has an old text file `movieData.txt` which contains the names and year(s) a movie was made (and remade).

```
...
A Tailor Made Man <1931>
A Tale of Five Cities <1951>
A Tale of Two Cities <1935> <1958> <1980> <1984>
A Talent for Murder <1984>
A Talk in the Dark <1992>
A Tall Man Executes a Jig by Irving Layton <1986>
...
```

Bob would like to be able to interactively search for all movies containing specified word(s) in the title or made during a specific year.

The program's main menu should contain the following options:

- start a new search - allows the user to enter one or more words in the title to search and/or year of the movie. This should just report the number of movies that match (he might not want to see 100s of movie titles). Multiple search words ALL must appear in the titles found.
- refine the search - allows the user to enter one or more additional words to refine the search. This should also report the number of movies that match.
- display to the screen the results of the current search: complete movie titles and the year(s) they were made
- print the results of the current search to a user-specified text file

At the start of the program, you should create an empty `ChainingDict` dictionary and fill it by reading movie entries from the `movieData.txt` file. Each dictionary entry will have a word for its key and a list of movie entries containing that word in their titles. (There are 34083 unique words if you split the movie titles on white-space, so having 40000 slots in the hash table should be about right, i.e., `ChainingDict(40000)`.)

NOTE MAC USERS: You'll want to open the `movieData.txt` file using:

```
movieFile = open('movieData.txt', 'r', encoding='utf8')
```

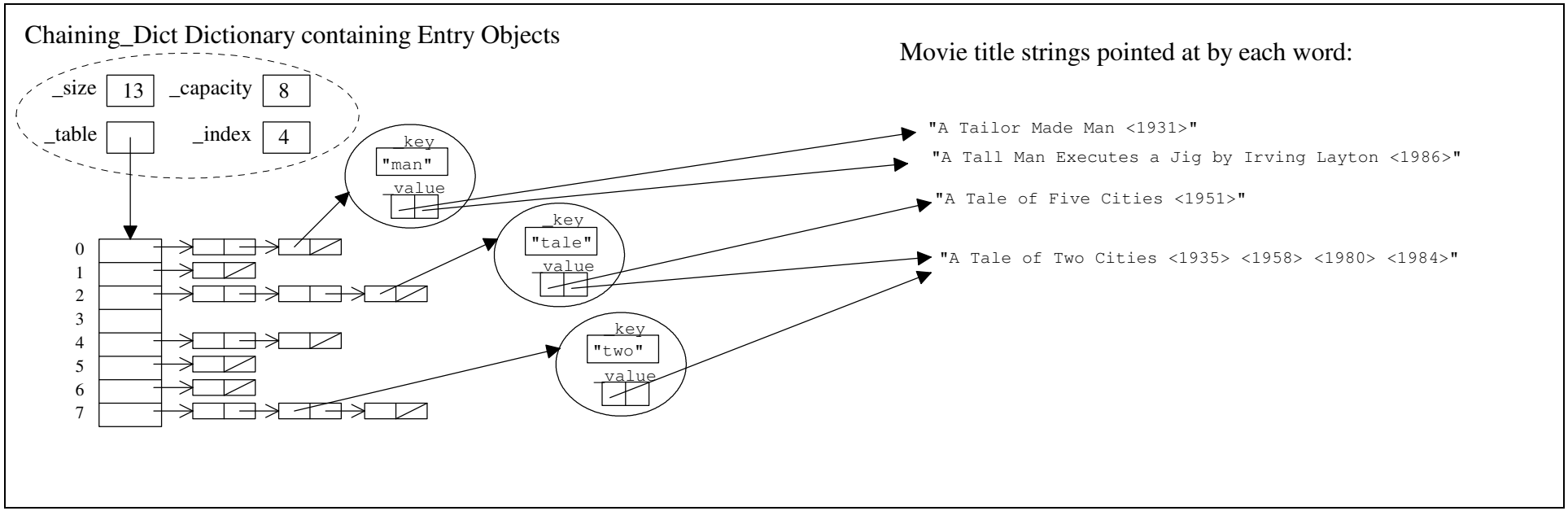
On a new search, use the longest word to search the dictionary for a list of movies. Use this list to find the matches if other search words were entered.

Design in Advance: Part of your grade will be determined by how well your code is split into functions. Thinking about design before coding can really help here.

Implement AND fully test your movie-search program. Part of your grade will be determined by how robust your program runs (i.e., does not crash) and how user-friend/intuitive your program is to use.

Submit ALL necessary files (chaining_dictionary.py, entry.py, moiveData.txt, etc.) with your movie-search program file(s) as a single zipped file (called hw4.zip) electronically at

https://www.cs.uni.edu/~schafer/submit/which_course.cgi



EXTRA CREDIT Options:

- More ellaborate searching options. Currently we "AND" together the search words, i.e., the titles found must contain all of the words. Other options might be to "OR" together search words, or allow some way to exclude titles with specified words.
- Provide a GUI front-end to the search program using Tkinter or other packages. A good tkinter references are at:
  http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html

  http://docs.python.org/2/library/tkinter.html