CS-UY 3254 — Homework 1

Jeff Epstein

1 Introduction

In order to help familiarize you with the Go programming language, you are asked to complete a short assignment that draws on your previous programming experience.

2 Assignment

Six Degrees of Kevin Bacon is a game that attempts to show that the actor Kevin Bacon is at the center of the universe. For this project, we will limit our scope to all movies ever made, and thus for our purposes, the game attempts to show merely that Kevin Bacon is at the center of the movie universe.

For all actors A, we define a "Kevin Bacon number," (KBN) which is defined like this:

- If A is Kevin Bacon, A has a KBN of zero.
- If A was in a movie with Kevin Bacon, A has a KBN of one.
- Otherwise, A has a KBN of $n + 1$, where n is the KBN of actor B, where B has been in a movie with A, and B has the lowest KBN of all actors who have appeared in a movie with A.

Another way of thinking about Kevin Bacon numbers is to imagine actors connected by movies. A connection exists between two actors if they were in a movie together. An actor has a finite Kevin Bacon number if there is a path from that person to Kevin Bacon, formed by traversing the connections. The value of that actor's Kevin Bacon number is equal to the length of the shortest such path. If there is no such path, we can say that the actor's Kevin Bacon number is infinity.

For the mathematicians in the audience, a Kevin Bacon number is similar to an Erdős number, but with movies instead of publications.

Your program should repeatedly prompt the user for an actor's name. If the given actor doesn't exist, an error message should be displayed. Otherwise, your program should output the list of actors and movies that form the shortest connection between the given actor and Kevin Bacon. (There may be more than one distinct shortest path; pick one.) Finally, your program should output the length of the path, which is also the Kevin Bacon number of the given actor. Your program will end when the user enters a blank line.

Here is a sample run of the program. The output of your program should be in an identical format.

```
Enter actor name: Josef Vetrovec
Josef Vetrovec was in Andel sv?d? d?bla (1988) with Michaela Vitkova
Michaela Vitkova was in Utz (1992) with Peter Riegert
Peter Riegert was in Animal House (1978) with Kevin Bacon
Found with KBN of 3

Enter actor name: Foobar Smith
Unknown actor name
```

```
Enter actor name: Madonna (I)
Madonna (I) was in League of Their Own, A (1992) with Tom Hanks
Tom Hanks was in Apollo 13 (1995) with Kevin Bacon
Found with KBN of 2

Enter actor name: Grace Ariyawimal
Grace Ariyawimal was in Visidela (1994) with W. Jayasiri
W. Jayasiri was in Mother Teresa: In the Name of God's Poor (1997) with Alan Shearman
Alan Shearman was in Jake Speed (1986) with Ken Gampu
Ken Gampu was in Air Up There, The (1994) with Kevin Bacon
Found with KBN of 4

Enter actor name: Yuriy Sak
Infinite KBN

Enter actor name: Yamil Turk
Yamil Turk was in Interior de un silencio, El (2005) with Andr?s Su?rez
Andr?s Su?rez was in 3 historias de juguete(s) (2005) with Gisel Wojtasek
Gisel Wojtasek was in Inc?modos (2006) with Juan Manuel Tellategui
Juan Manuel Tellategui was in Puto (2005) with Cesar Casalone
Cesar Casalone was in Mataperros (2002) with Diego Capusotto
Diego Capusotto was in Regresados (2007) with Diego Leske
Diego Leske was in Highlander II: The Quickening (1991) with Allan Rich
Allan Rich was in Hero at Large (1980) with Kevin Bacon
Found with KBN of 8
```

I've provided a movie database in a file named `cast.txt` that provides information about the relationships between movies and actors. It contains all movies from IMDB, the Internet Movie Database web site. Its format should be clear by looking at it. Note that some actors' names are suffixed with a parenthesized Roman numeral to distinguish actors of the same name. Non-ASCII characters in names and titles have been replaced with question marks in order to simplify text processing. When your program starts, it should load this file automatically. It should not prompt the user for a file name. When the file is loaded, your program can prompt the user for the first actor to search for. All queries should be performed in memory, without the need to refer to the file again.

In order to get full credit, your program should be able to calculate the KBN for any actor more-or-less instantaneously.

3 Rules

Operating system I will evaluate your program submission by running it under the GNU/Linux operating system, in particular a Debian or Ubuntu distribution. Your grade will therefore reflect the behavior of your project code when executed in such an environment.

While you are welcome to develop your project under any operating system you like (such as Windows or Mac OS), you are responsible for any operating system-dependent deviations in program behavior, including but not limited to variations pertaining to file path separators, thread synchronization, network sockets, and file permissions. In order to assure yourself of getting the highest possible grade, you are encouraged to test your project in a Linux environment, even if Linux is not your primary operating system. You may want to use a Linux virtual machine.

Language You should implement this project in Go. Please use at least version 1.10. Use the `go version` command to check the the currently installed version.

Libraries You are free to make use of all packages of the Go standard library (that is, all packages that are installed by default with Go).

If you wish to use any other third-party packages, please check with me first before starting. I will likely approve such a request if you can make a persuasive case that a particular library will improve your project. Without prior approval, use of external libraries and frameworks is prohibited.

Academic integrity You should write this assignment entirely on your own. You are specifically prohibited from submitting code written or inspired by someone else, including code written by other students. Code may not be developed collaboratively. You may rely on publicly-accessible documentation of the Go language and its libraries. Please read the syllabus for detailed rules and examples about academic integrity.

Code quality You should adhere to the conventions of quality code:

- Indentation and spacing should be both consistent and appropriate, in order to enhance readability.
- Names of variables, types, fields, and functions should be descriptive. Local variables may have short names if their use is clear from context.
- All functions should have a comment in the appropriate style describing its purpose, behavior, inputs, and outputs. In addition, where appropriate, code should be commented to describe its purpose and method of operation.
- Your code should be structured to avoid needless redundancy and to enhance maintainability.

In short, your submitted code should reflect professional quality. Your code's quality is taken into account in assigning your grade.

If I think that the quality of your code (independent of the correctness of your program) is not of professional quality, I may ask you to make corrections and resubmit your work before I will give you a grade.

4 Hints

- In Debian or Ubuntu Linux, you can install Go with this command:

  ```
  sudo apt-get install golang
  ```

  You will need to set your GOPATH to point to a directory where you will keep your Go sources, libraries, and binaries. I propose the imaginative name of `godir` for that directory. Create it and set GOPATH like this:

  ```
  mkdir -p godir/src
  export GOPATH=$(realpath godir)
  export GOBIN=$(realpath godir)/bin
  ```

  Now let's create a file in that directory:

  ```
  gedit godir/src/test.go
  ```

  Give it the following content:

```
package main

import "fmt"

func main() {
    fmt.Println("Hello world")
}
```

Now compile the file like this:

```
go install godir/src/test.go
```

This will produce an executable file named `test` which you can run like this:

```
godir/bin/test
```

- Go has excellent documentation online. See `https://golang.org/doc/`.
- To read a line from the keyboard, in a manner similar to Python's `input()` function, I recommend the following incantation:

```
line, err := bufio.NewReader(os.Stdin).ReadString('\n')
if err != nil {
    .... // handle error here
}
line = line[:len(line)-1]
```

The last line above is necessary to remove the newline character from the end of the input string.

Note that you will need to `import` the `bufio` and `os` modules in order to use the above code.

- You may find it useful to take advantage of the Go's `map` type, which is equivalent to Python's `dict`.

5 Submission

Submit your work on Gradescope. Remember to submit all source files necessary to build and run your project. Do *not* submit binary executable files: please use the `go clean` command to remove unnecessary object files before submission. Do not submit the data file. If your project consists of more than one source file, submit them all together in a zip file that preserves the relevant directory structure.