User Manual

# Assignment 1: Random Sentence Generator

**Topics:**

UNIX, command line, Makefile, compilation process, STL map, recursion.

## How to Run the Code:

To run and test my solution to Assignment-1, please run the following commands at the terminal:

$ cd Assignment\_1

$ cd assn-1-rsg

$ make

$./rsg-sample-linux ./assn-1-rsg-data/grammarFileName

where grammarFileName is a file from the folder assn-1-rsg-data. For example, to test the grammar file poem.g, please type:

./rsg-sample-linux ./assn-1-rsg-data/poem.g

The output would be 3 random expansion of the pertaining grammar, for instance:

Version #1: ---------------------------

The waves die warily tonight.

Version #2: ---------------------------

The waves sigh warily tonight.

Version #3: ---------------------------

The big yellow flowers portend like waves tonight.

# Assignment 2: Six Degrees of Kevin Bacon

## Topics:

Pointers, void \*, pointer arithmetics and casting, memory, C++ classes and templates, to see a legitimate scenario where a complex program benefits from two types of paradigms: high-level C++ (with its templates and its object orientation) and low-level C (with its exposed memory and its procedural orientation.)

## How to Play:

The game takes the form of a trivia challenge. Propose two names, and your friend / opponent has to come up with a sequence of movies and mutual co-stars connecting the two. In this case, your opponent takes on the form of your computer, and the computer is exceptionally good. Example: Barry Manilow and Lou Rawls:

Actor or actress [or <enter> to quit]: Barry Manilow

Another actor or actress [or <enter> to quit]: Lou Rawls

Barry Manilow was in "Bitter Jester" (2003) with Dom Irrera.

Dom Irrera was in "Man Is Mostly Water, A" (2000) with Lou Rawls.

## Overview

There are two major components to this assignment:

1. You need to provide the **implementation for an imdb class1,** which allows you to quickly look up all of the films an actor or actress has appeared in and all of the people starring in any given film. You’ll tap your sophisticated understanding of memory and data representation in order to look up movie and actor information very, very quickly.
2. You also need to implement a **breadth-first search algorithm** that consults your super-clever imdb class to find the shortest path connecting any two actor/actresses. If the search goes on for so long that you can tell it’ll be of length 7 or more, then you can be reasonably confident (and pretend that you know for sure that) there’s no path connecting them. This part of the assignment is more CS106B-like, and it’s a chance to get a little more experience with the STL and

**How to Run the Code:**

to compile and run the solution, please type the following commands on the terminal:

$ cd Assignment\_2

$ cd assn-2-six-degrees

$ make

$ ./six-degrees

Once ran, the program asks for two artist names to play.

# Assignment 3: Vector

**Topics:**

write a few generic container data structures to imitate the functionality of the STL—without using templates, memory access and management, generic C functions, C library functions: malloc, realloc, free, memcpy, memmove, qsort, and bsearch, C structs.

## Overview

Implement a generic C vector and C hashset, and test them using vector-test and hashset-test.

**How to Run the Code:**

To run and test the code, type the following functions:

$ cd Assignment\_3

$ cd assn-3-vector-hashset/

$ make

$ ./vector-test

$ ./hashset-test

**Assignment 4 and 6: RSS News Feed Aggregation**

generic containers, to build a fast, super-lean, industrial-strength application, multi-threading and semaphores

**How to Run the Code:**

Some of the resources (C libraries) necessary for assignments 4 and 6 require a Stanford Network Account and therefore are not accessible to us. With my hands being tied for compiling the base code, I have written an algorithm as below:

main{

open the feeds file

initialize database

initialize thread package

set up three semaphores : PreviouslySeenArticlesOpen, indicesOpen, stopWordsOpen as members of the database

Welcome {

read and print the contents of the welcome file

}

LoadStopWords {

read the stopWords from file,and store them in the database's stopWords hashset

}

BuildIndices {

Initialize database's indices and previouslySeenArticles

initialize semaphore rssFeedAcess(4) to manage number of rssFeeds

for each rss feed listed in the file {

make a thread to go through rssFeed as below

processFeed {

if a connection to the feed can be made {

semaphoteWait(rssFeedAccess)

PullAllNewsItems {

initialize Semaphore urlconnOpen(24) to limit number of open urls to 24

for each url in the feed {

processStartTag {

define a new rssFeedEntry and add title and url

}

ProcessEndTag {

make a thread to parse the articleand pull the indices as bellow{

parseArticle {

make a connnection to the news article

semaphoreWait(PreviouslySeenArticlesOpen)

add to the previouslySeenArticles,

semaphoreSignal(PreviouslySeenArticlesOpen)

semaphoreWait(urlOpen)

ScanArticle{

for each word in the article {

AddWordToIndices{

if the wordIsWortIndexing, add it to indices,

useing indicesOpensemaphore

}

}

}

semaphoreSignal(urlOpen)

}

}

}

}

}

semaphoteSignal(rssFeedAccess)

}

}

}

}

run all threads

QueryIndices {

allow user to sepecify a single search term

}

processResoponse {

list 10 relevant articles

}

SemaphoreFree {

free all the semaphores

}

}