COMP 206 Winter 2018 – Assignment 3

VERSION B – March 27th, 2018

(change since A: added another possible option for getting started 3c that some people needed)

Objectives:

To provide a light-weight experience with several of the software systems concepts of the past few weeks: Makefiles, Git, C structures, C function pointers, and bash scripting.

NOTE: This Assignment is meant to take *much* less time than the other 3. I estimate you should be able to complete it in 2 hours max, so use this in your time management and do post on My Courses right away if you get stuck on something.

Getting started:

The A3 repository is at: https://github.mcgill.ca/COMP206Winter2018/Assignment3. Create your own mirror to start working on the code:

1. Login to https://github.mcgill.ca/, click the green "New Repository" button
2. Enter repository name "Assignment3_submission" and choose "Private". Press green "Create repository" button. You should now see an empty repo.
3. In a terminal, type the following commands to get a mirror of the A3 files:
    a. $ git clone --bare git@github.mcgill.ca:COMP206Winter2018/Assignment3.git
        i. If you get any permissions error here, remember you might need to add new SSH keys here: https://github.mcgill.ca/settings/keys
    b. $ cd Assignment3.git
    c. $ git push git@github.mcgill.ca:<YOUR_USERNAME>/Assignment3_submission.git
        i. Alternative, if you get an error: "$ git push git@github.mcgill.ca:<YOUR_USERNAME>/Assignment3_submission.git master"
    d. $ cd ..
    e. $ rm –rf Assignment3.git
    f. $ git clone git@github.mcgill.ca:<YOUR_USERNAME>/Assignment3_submission.git

You will now have the folder "Assignment3_submission" in your working directory, and if you refresh your GitHub page, you'll see the files there also. This is the place to do all your work.

Handing In:

This assignment will be graded primarily by a TA who inspects your Github repository. However, for record keeping purposes, and to help us find your repo, you'll also submit an archive to My Courses.

Just before submitting, create a file called “repo_path.txt” in the top level of your repository. Ensure it includes one single line, the URL of your GitHub repository. For example, mine would say: "https://github.mcgill.ca/david-meger/Assignment3_submission".

Create an archive by running "$ tar czf Assignment3.tar.gz Assignment3_submission" from one level above your repository folder. Submit the Assignment3.tar.gz to My Courses.

Deadlines:

1. One single deadline, Wednesday April 4th (Note: just one week, since this is small and we need to fit in A4 which will be a little larger.)

Intro: 206-flix

One of the most important operations for a software system these days is to analyze information about a large collection of users. This can be for reasons in the news right now like influencing politics, but it is often for really helpful reasons like finding us good songs and movies we're likely to enjoy, based on the recommendations of similar people.

In this assignment, we will look at a very simple prediction engine, designed to recommend movies based on your answers to the 13 “silly questions” that we collected in Assignment 0. We’ve provided you all of the necessary C code and data. You will just need to read and understand this code and then write a few supporting software system components so the code can be built and can make predictions for everyone in the class.

Question 1: Track your progress with Git (20 marks)

If you completed "Getting Started", you now have your own GitHub repo and local copy checked out. You must continue to use Git while you complete the other tasks. Make at least 5 commits that are tracked and have reasonable log messages (the number of commits will show up on "code" tab of your GitHub page and starts at 2 which are made by me. Yours should say 7 minimum when you're done). It’s not necessary to branch, merge or any other complex operation, but feel free to try this if you’d like.

The workflow you'll probably find most convenient for making a commit is:

1. Use your text editor to create or modify some files in the repository. Save them.
2. Run "$ git add <filenames>" which tells git the changes exist.
3. Run "$ git commit –m <log_message>" which tells git you want the changes to stick, and describes them with your log message.
4. Run "$ git push", which shares the commit from your local folder to GitHub. After this, you should see the files changed, and the commit counter should increase.

Question 2: Code Reading (30 marks)

The first important thing is for you to understand the provided code. It uses structs, function pointers and pre-compiler macros to deal with the various types of data that were entered to answer the questions. You should open the C files in the src and include directories, trace through and understand how they work, and then answer the following 3 questions with 5 lines of text maximum each:

1) Explain the overall function-call structure of the program. Starting with main, which functions are called and in what order. Include those abstracted through function pointers and pre-compiler macros, but stop before you get to C built-ins like strcpy or atoi.
2) Consider the **LOAD_FIELD** function, which converts all data from the user’s text input into the struct’s fields. Explain how this single function can handle various types of data.
3) Consider the global integer array **field_offsets.** Explain why this is the correct way to compute the offset to each field in the struct, based on what we learned in class. It may be helpful to check the documentation of **offsetof** here: http://en.cppreference.com/w/c/types/offsetof.

Write your answers in the README.md file located in the git repository. Use mark-down syntax, including at least 2 levels of header and 1 list. Instructions here: https://guides.github.com/features/mastering-markdown/

Question 3: Create the Makefile (30 marks)

The project currently has no Makefile, and therefore must be built manually with the command:

$ gcc -Iinclude src/movie_recommender.c src/preferences.c src/distances.c -o movie_recommender

(Note the first option is read out "minus capital eye", which tells gcc to look for .h headers in the directory called include.)

Add a Makefile to the repository (use git add, commit, push). When complete, simply typing "$ make" in the directory must produce the file "movie_recommender", recompiling it only if one of the .c or .h files has changed. You are not meant to change any C files for this assignment, but you can test with "$ touch src/preferences.c" for example, which simply updates the modification time, and triggers recompile Typing "$ make clean" must then delete this file. You are allowed to create the simplest Makefile that accomplishes this and are not obliged to use any make macros or complex features.

Question 4: Create generate_all_predictions.bash (20 marks)

After building, the code can be run with the following example:
$ ./movie_recommender query/daves_preferences.txt data/*
Top movie pick for user David Meger is: Brooklyn.

The program's first argument is a filename for the target user (the one who needs a recommendation) and all other files listed are considered "data", which are mined to make a good suggestion. Note that the "*" character expands to give your program every filename in the data directory. Because the prediction is based on finding the "nearest" user, it's important that the target is not also listed in the "data" list, otherwise they will always just be told to watch their own favorite movie.

Write "generate_all_predictions.bash" that generates predictions for all users looping over all files in the data directory and treating them one by one as the "target". For each target:

1. Move the target out of the data directory and into the query folder
2. Run the movie_recommender program as shown
3. Move the target back into the data folder so it can be used next time.

Slide 43 of Lecture15 is a good resource to get started here. When complete, your script should be around 5-10 lines long and should get 205 lines out output that look as follows:

$ bash generate_all_predictions.bash
Top movie pick for user Benjamin Labrecque is: Spirited Away.
Top movie pick for user Susan Matuszewski is: Ile Flottante.
Top movie pick for user Brendan Furtado is: La La Land.
Top movie pick for user Andy Zhen is: .
Top movie pick for user Andy Zhen is: .
Top movie pick for user Shayan Sheikh is: Brooklyn.
Top movie pick for user Kathy Tam is: About Time.
Top movie pick for user Yuxue Zhu is: Across the the Universe.

Note: The code is imperfect and some users have entered their preferences in slightly different formats. So, you may see segmentation faults and core dumps occasionally. This is not your responsibility to fix, just ignore them as long as you get mostly good predictions.