Due: - Written Exercises due Monday, May 16th, at 4 pm in Homework box in 2131 Kemper;
        - Program due electronically Monday, May 16th, at 11:50 pm.

Written Exercises (20 pts): The written exercises should be typed and each page should have at the top your name and ID#, section, and hw#. Handwritten answers will not be graded.

J&K, 7.6.4, 7.6.6, 7.6.8, 11.1.2, 11.1.4, 11.1.6, 11.1.8, 11.2.4.

Program (80 pts): Handin a makefile and source code files, and have your makefile produce the executable file **recommender** (use **all** ... at the top). The third line in the source code files must contain the author of the file, ID, and section #. Use the *handin* program for electronic submission, described in the UNIX tutorial. For this homework use:

handin cs30 hw6 Makefile <your file1> <your file2> ...

The date and time your files are created in the cs30 directory will be counted as your submit times. If those times are later than 11:50 pm on the due date your submissions will be considered late.

Open Source Software Recommender System

Open Source Software (OSS) are large software systems produced by communities of developers, and their source code is generally publicly available. Examples are the Apache web server, the Linux operating system, and others. Because these systems are large (millions of lines of code across hundreds or even thousands of functions) and since the work load in an OSS project is distributed among many people, tools exist to help the community deal with the complexity of adding to or changing the software. One class of such tools are **recommender systems** which help developers mine the existing software code to find related code of interest. In particular, recommender systems exist to recommend related functions to a given C function on which a developer might be currently working. Using such recommendations can be invaluable when a developer is changing the code of a C function in a way that then affects other functions (e.g. changing a function's number of parameters or their type). Here you are asked to write a simple recommender system which only looks at the overlap between function names, and recommends a function with the longest name overlap to a given function.

Write a program that, given a user specified file of function names and a separate user specified function name, recommends a function name that has the longest substring overlap with the specified function name. The program must first ask the user for the name of a file which contains function names. Assume the function names are strings, of length at most 50 characters, and no white spaces. Each line of the input file contains a single function name. Assume there are at most 200 functions in the file. The program should then read in the function names from this file and ask the user for a function name. If the function name exists in the file, the program must output another function name from the file with the longest substring overlap. If no overlap > 0 is found, make no recommendation. **Your program must use ragged arrays for storing the names of the functions, and not 2d arrays. Any implementation without ragged arrays will get less than 50% of the points even if correct.**

Make sure the output of your program matches exactly the output below of my executable located at **/home/cs30/public/hw6/recommender** on the csif machines. In the same directory you will find files **apache_small.txt** and **apache_big.txt** which contain example function names.

```
[cs30@pc50 hw6]$ less apache_small.txt

make_sub_pool
clear_pool
destroy_pool
bytes_in_pool
bytes_in_free_blocks
palloc
pcalloc
pstrdup
pstrndup
pstrcat
make_array
push_array
zzz

[cs30@pc50 hw6]$ ./recommender
Name of network file: apache_small.txt
Function name: make_array
Recommended function: push_array

[cs30@pc50 hw6]$ ./recommender
Name of network file: apache_small.txt
Function name: make_cake
There is no function make_cake in file apache_small.txt

[cs30@pc50 hw6]$ ./recommender
Name of network file: apache_small.txt
Function name: zzz
No recommendation.
```