# CS 3352 – Proj1
# A Project to gain C programming experience on UNIX systems
# Due – Feb. 20, 2010 at 11:59 pm

**Problem**: I want to recommend the top two students in my class for some scholarship awards. These students should have the GPA of at least 0.5 more than the average of the whole class. The student data, the input for this project, is stored in the file, P1-DATA, of the directory "/home/courses1/cs3352/projects/proj1". It is a text file with each line containing the first and the last name, student-id, and the current GPA of a student in the class. Your job is to develop a C program to identify the top two students for my recommendation.

Your program should obviously read the input file twice, first for computing the average GPA of the whole class and second for determining the top two students who meet the selection criterion. It should store the entries of the selected students in two structures and finally call a procedure to print the entries. **The output should be a table of four columns, with each column having appropriate column heading.** Your program should contain at least two procedures, main() and my_print() (or whatever you name to your print procedure). Develop this project in a separate folder.

An important purpose of this project is to learn C program development environment. Hence, this project is divided in two stages: In stage 1, you will implement the given problem in a single source file; In stage 2, you will divide the program of stage 1 into two files, create a library of modules and link them using linker/loader.

**Stage1**: (1) First visit "/home/courses1/cs3352/projects/proj1/" and view "P1-DATA". **Do *not* copy this file to your proj1 directory, because your program should read data directly from P1-DATA.**

(2) Each input line begins either with a space or '#' and contains a student data (two strings, an integer, and a decimal number). A line that starts with "#"indicates that the student is already holding some other scholarships. GPA of such students is accounted for calculating the class average, but these students are not to be selected for my recommendation.

(3) Next, read the man pages of fopen ( ), freopen ( ), fgetc ( ), fgets(), sscanf() and feof ( ). Then study "file_use.c" stored in "~cs3352/tutorials" and follow it to develop your main program. Replace "file_input" by the full path name of the input file.

The following code shows how you can read an input line in two parts, the first character and then the rest of the line.

```
while (1) // 1 is always non-zero, hence there is an infinite loop
  {if (foef (…) break; //quit loop when end-of-file reached
    Use fegtc() to read the first character of a line.
    Use fgets() to read the rest of the line and save in a character array.
    Use sscanf() to fetch four pieces of an input line from the array.
    Process data;
  }
Compute and print the average GPA.
```

Note that you should print all output on the screen. If you rather save it in a file, the file MUST be created in the current directory, i.e., you must not use your HOME pathname in the fopen(). Remember this caution for future assignments.

To test if your program reads the input file fully and correctly, print the four pieces of data from each line. Do NOT store the input in an array; the code in "file_use.c" does show syntax of an array of structures, but it is given to teach additional C code.

(4) There are two ways to read the input file twice: (a) close the file after reading all lines, and open it again, or (b) use freopen ( ); in this case, you should not close the file to read it back. Next repeat the above loop to process the second round.

(5) Define a structure, say STUDENT, to hold student input and declare two variables of this structure, say FIRST and SECOND. During the second round, if a student meets the award requirement and his GPA is higher than the GPA of the entry stored in FIRST or SECOND, save the current student input in the structure that has smaller GPA.

(6) Code my_print(). Its argument is a pointer to the STUDENT structure. Call this procedure twice from the main program. This procedure should print the four items of a student record in a line. Use appropriate format code so that output are aligned in each column.

(8) Compile as follows: "gcc stage1.c –o STAGE1" (assuming that you named your source file "stage1.c" and you want the executable to be called "STAGE1"). Run "./STAGE1". Note that the input filename is hard coded; hence, there are no arguments.

It is advised that you edit ".bashrc" file and add "." (the current directory) to the PATH environment variable by appending ":.". Run "source .bashrc". You would then be able to run without typing "./", i.e., "STAGE1" should be sufficient to execute it.)

**Stage2**: Building own library.

(9) Make a copy of stage1.c to stage2.c. Next, move the definition of STUDENT from stage2.c to a new file, say "my_header.h". Also move "my_print ( )" to a separate file, say "output.c". You still pass the address of the structure as an argument to my_print ( ). See my file "struct_use.c", and follow its code. Include my_header.h to both stage2.c and output.c.

**(10)** Compile stage2.c and output.c separately as below. "-c" option tells the compiler to just compile the source files and not call linker/loader to create the final executable file.
gcc –c *.c

Next prepare an archive, say, "libMy.a" using the archive command "ar" as follows and place output.o in your library.

/usr/ccs/bin/ar –r –v libMy.a output.o

Run the following to see its listing:
/usr/ccs/bin/ar –t libMy.a

run "ld" to produce executable "STAGE2" (output.o will be fetched from libMy.a).

ld –o STAGE2 stage2.o –lMy –lc

ld is the linker program. "-lc" denotes the library libc.a (the standard library of C language). Note that the linker assumes that every library name begins with "lib"; hence library libXXX.a is specified by –lXXX on the "ld" command line. Place your library on LD_LIBRARY_PATH so that ls

Run STAGE2.

**If the loader gives a message such as libMY not found, your LD_LIBRARY_PATH is not set properly. You can do one of the two things. Either set LD_LIBRARY_PATH in your .bashrc or provide the path in the command line.**

ld –o STAGE2 proj1.o  -L. –lMy –lc

In the above command line,  –L(dot) tells “ld” to search the current directory for additional object modules.

To modify LD_LIBRARY_PATH, add at the end of this path in your .bashrc, “:$HOME/cs3352/proj1” (assuming that libMy is stored in your cs3352/proj1 directory).

The correct output is saved in the file P1-OUT in the same folder of the input file.

**Learning outcome:** Since you are given a complete design and plenty of C code, you might not recognize all what you learnt through this project.  Therefore, after completing the project, you should prepare a list noting down all C constructs, library functions, and development tools you used in this project. This is like reviewing the project. Note that in the next project, you might need what you learnt in this project. Therefore, preparing such a list would be very helpful.