ECS 40 **Program #2 (45 pts)** **Winter 2016**

Due: Wednesday, January 27th, 11:59pm
Use handin to submit authors.csv, Ins.c, NoComments.sh, HeadFiles.sh, stdio.sh, and makemake.sh to the p2 directory in cs40a. All files, except Ins.c, must be submitted from only one account of the team.

Format of authors.csv: author1_email,author1_last_name,author1_first_name
author2_email,author2_last_name,author2_first_name
For example: simpson@ucdavis.edu,Simpson,Homer
potter@ucdavis.edu,Potter,Harry

ddd Tutorial (10 points)

Follow the directions of the DDD tutorial available online at http://heather.cs.ucdavis.edu/~matloff/Debug/Debug.pdf. Each person must do the tutorial individually. The authors.csv for the partner that is only submitting Ins.c should contain only one name. You will find Ins.c, and Lnk.c in ~ssdavis/40/p2. When done completely debugging Ins.c, handin it.

There are at least four ways to gain access to ddd:

1. Go to the basement of Kemper and select DDD Debugger from the Programming menu.
2. To use ddd at home under Windows, on programs developed at home.
   - 2.1. Install cygwin (available for free from cygwin.com) with g++, openssh, and ddd. The selection of ddd should automatically install the X windowing server for cygwin.
   - 2.2. Once cygwin is installed, type **xinit&** at the cygwin command prompt to open an X window, and then type **ddd** at the prompt.
3. To use ddd at home under Windows, on programs developed in the CSIF.
   - 3.1. Install cygwin with at least the X server (I would still suggest installing g++ and ddd).
   - 3.2. Once cygwin is installed, type **xinit&** at the cygwin command prompt to open an X window.
   - 3.3. Type ssh –X username@CSIF_computername
   - 3.4. Once you have logged into the CSIF computer, change to the appropriate directory, and then type **ddd&**
4. To use ddd at home under MacIntosh OS X, on programs developed in the CSIF
   - 4.1. Open an X term. (See the MacIntosh help to install the X package)
   - 4.2. Type **ssh –X username@CSIF_computername**
   - 4.3. Once you have logged into the CSIF computer, change to the appropriate directory, and then type **ddd&**

Bash Shell Scripts (35 points)

Each script must use the bash shell, so use "#! /bin/bash" as the first line. A good tutorial is at http://steve-parker.org/sh/sh.shtml

1. (5 points, 10 minutes) Name: NoComments.sh
   You have a number of C programs that contain comment lines that begin with /* followed by the first line of comment, but the terminator line has */ as the *only* characters in the line. Remove these comments from all of the .c files in the current directory.

```
[ssdavis@lect1 p2]$ mkdir temp
[ssdavis@lect1 p2]$ cp Testing/NC/* private/NoComments.sh temp
[ssdavis@lect1 p2]$ cd temp

[ssdavis@lect1 temp]$ cat NC1.c
First line NC1.c
/* Comments
Second line NC1.c
  */
*/
adsf
asfd
asdf
asdf
asdf

[ssdavis@lect1 temp]$ cat NC2.c
/* First line of NC2.c
asdf
asdf
*/
asdf
asdf
asdf

[ssdavis@lect1 temp]$ cat NC3.c
First line of NC3.c
as/*df
asdf
*/
asdf
asdf
asdfasdf
asdf

[ssdavis@lect1 temp]$ NoComments.sh
[ssdavis@lect1 temp]$ cat NC1.c
First line NC1.c
adsf
asfd
asdf
asdf
asdf

[ssdavis@lect1 temp]$ cat NC2.c
asdf
asdf
asdf

[ssdavis@lect1 temp]$ cat NC3.c
First line of NC3.c
as/*df
asdf
*/
asdf
asdf
asdfasdf
asdf

[ssdavis@lect1 temp]$
```

2. (5 points, 10 minutes) Name HeadFiles.sh
   Write a script that behaves both in interactive and non-interactive mode. When no arguments are supplied, it picks up each .txt file in the current directory, lists its first 3 lines, and then prompts for deletion of the file. If the user supplies arguments with the script, it works on those file only.

```
[ssdavis@lect1 temp]$ cp ../Testing/HF/* ../private/HeadFiles.sh .
[ssdavis@lect1 temp]$ ls
HeadFiles.sh  HF1.txt  HF2.txt  HF3.c  HF4.txt  HF5.txt  HF6.csv
[ssdavis@lect1 temp]$ HeadFiles.sh HF2.txt HF3.c HF5.txt
Displaying first 3 lines of HF2.txt:
HF2 first
HF2 second
HF2 third

Delete file HF2.txt? (y/n) y
Displaying first 3 lines of HF3.c:
HF3 first
HF3 second
HF3 third

Delete file HF3.c? (y/n) n
Displaying first 3 lines of HF5.txt:
HF5 first
HF5 second
HF5 third

Delete file HF5.txt? (y/n) y
[ssdavis@lect1 temp]$ ls
HeadFiles.sh  HF1.txt  HF3.c  HF4.txt
HF6.csv
[ssdavis@lect1 temp]$ HeadFiles.sh
Displaying first 3 lines of HF1.txt:
HF1 first
HF1 second
HF1 third

Delete file HF1.txt? (y/n) n
Displaying first 3 lines of HF4.txt:
HF4 first
HF4 second
HF4 third

Delete file HF4.txt? (y/n) y
[ssdavis@lect1 temp]$ ls
HeadFiles.sh  HF1.txt  HF3.c  HF6.csv
[ssdavis@lect1 temp]$
```

3. (5 points, 10 minutes) Name: stdio.sh
Write a script which looks up every .c file in the current directory for the strings printf or fprintf (note that sprintf, among others, does not match these patterns). If found, and the file does not already have `#include <stdio.h>` in it, then the script adds the statement `#include <stdio.h>` at the beginning of the file.

```
[ssdavis@lect1 temp]$ rm *
[ssdavis@lect1 temp]$ cp ../Testing/stdio/* ../private/stdio.sh .
[ssdavis@lect1 temp]$ find . -name "*.c" -print -exec cat {} \;
./S1.c
asdf
printf("what");

./S2.c
asdf
fprintf("what");
asdf

./S3.c
#include <stdio.h>
printf("what");
asdf

./S4.c
First line
sprintf("what");
asdf

./S5.c
First Line
#include <stdio.h>
asdf
asdf
printf("What")

[ssdavis@lect1 temp]$ stdio.sh
[ssdavis@lect1 temp]$ find . -name
"*.c" -print -exec cat {} \;
./S1.c
#include <stdio.h>
asdf
printf("what");

./S2.c
#include <stdio.h>
asdf
fprintf("what");
asdf

./S3.c
#include <stdio.h>
printf("what");
asdf

./S4.c
First line
sprintf("what");
asdf

./S5.c
First Line
#include <stdio.h>
asdf
asdf
printf("What")

[ssdavis@lect1 temp]$
```

4. (20 points) (30 minutes) Write a Bash shell script, makemake.sh, that will create a makefile called Makefile based on all the .cpp files in the current directory. If a .cpp file has any #includes of non-system header files (those with double quotes around them), then those files should be listed in its dependencies. The –Wall –ansi, and –g options will always be used with g++. The shell script takes the name of the executable as its first argument. If no argument is provided, the script should report the error, and print a usage statement. All other parameters are additional options that should be used with every call to g++. The Makefile should end with a "clean:" routine that removes the executable and all object files. (Hints: the -n option of echo inhibits the default newline, and \t and \n work with echo. I used sed and awk to get at the name of the header files within the .cpp files.)

```
[ssdavis@lect1 temp]$ cp ../Testing/MakeFiles/* ../private/makemake.sh .
[ssdavis@lect1 temp]$ ls
appointment.cpp  calendar.cpp   dayofweek.h  time.cpp  year.cpp
appointment.h    dayofweek.cpp  makemake.sh  time.h    year.h
[ssdavis@lect1 temp]$ [ssdavis@lect1 p2]$ ls
appointment.cpp  calendar.cpp  day.h          dayofweek.h  Lnk.c        private   time.h    year.h
appointment.h    day.cpp       dayofweek.cpp  Ins.c        makemake.sh  time.cpp  year.cpp
[ssdavis@lect1 temp]$ makemake.sh
Executable name required.
usage: makemake.sh executable_name
[ssdavis@lect1 temp]$ makemake.sh cal.out
[ssdavis@lect1 temp]$ make
g++ -ansi -Wall -g -c appointment.cpp
g++ -ansi -Wall -g -c calendar.cpp
g++ -ansi -Wall -g -c day.cpp
g++ -ansi -Wall -g -c dayofweek.cpp
g++ -ansi -Wall -g -c time.cpp
g++ -ansi -Wall -g -c year.cpp
g++ -ansi -Wall -g -o cal.out appointment.o calendar.o day.o dayofweek.o time.o year.o
[ssdavis@lect1 temp]$ make clean
rm -f cal.out appointment.o  calendar.o  day.o  dayofweek.o  time.o  year.o
[ssdavis@lect1 temp]$ makemake.sh cal.out -O2 -g
[ssdavis@lect1 temp]$ cat Makefile
cal.out : appointment.o calendar.o day.o dayofweek.o time.o year.o
        g++ -ansi -Wall -g -o cal.out -O2 -g appointment.o calendar.o day.o dayofweek.o time.o
year.o

appointment.o : appointment.cpp appointment.h
        g++ -ansi -Wall -g -c -O2 -g appointment.cpp

calendar.o : calendar.cpp year.h
        g++ -ansi -Wall -g -c -O2 -g calendar.cpp

day.o : day.cpp appointment.h day.h dayofweek.h
        g++ -ansi -Wall -g -c -O2 -g day.cpp

dayofweek.o : dayofweek.cpp dayofweek.h
        g++ -ansi -Wall -g -c -O2 -g dayofweek.cpp

time.o : time.cpp time.h
        g++ -ansi -Wall -g -c -O2 -g time.cpp

year.o : year.cpp year.h day.h
        g++ -ansi -Wall -g -c -O2 -g year.cpp

clean :
        rm -f cal.out appointment.o  calendar.o  day.o  dayofweek.o  time.o  year.o
[ssdavis@lect1 temp]$
```