ECS 40 **Program #2 (60 pts)** **Spring 2016**

Due: Wednesday, April 13th, 11:59pm

Use handin to submit authors.csv, Ins.c, cpdirs.sh, grepdir.sh, uncomp.sh, and makemake.sh to the p2 directory in cs40a.

cpdirs.sh, grepdir.sh, uncomp.sh, and makemake.sh should be submitted from only one account of the team. Since each team member will have their own Ins.c, a two-person team will submit two Ins.c, one from the account of each of its writers.

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.txt for the partner that is only submitting Ins.c should contain only one name. You will find Ins.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 (50 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. (7 points) Write a shell script, named grepdir.sh, that searches for a pattern in a directory, and all of its subdirectories. The starting directory is the first argument, the pattern is the second parameter, and the options for grep are all succeeding parameter(s). Options will start with a hyphen. The script should produce a usage statement if the script is misused.

```
[ssdavis@lect1 private]$ ls
cpdirs.sh  grepdir.sh  makemake.sh  temp  temp2  temp3  uncomp.sh
[ssdavis@lect1 private]$ grepdir.sh cpdirs.sh bin
usage: grepdir.sh directory pattern [-grep option]*
[ssdavis@lect1 private]$ grepdir.sh .
usage: grepdir.sh directory pattern [-grep option]*
[ssdavis@lect1 private]$ grepdir.sh cpdirs.sh bin
usage: grepdir.sh directory pattern [-grep option]*
[ssdavis@lect1 private]$ grepdir.sh .
usage: grepdir.sh directory pattern [-grep option]*
[ssdavis@lect1 private]$ grepdir.sh . bin
#! /bin/bash
#! /bin/bash
#! /bin/bash
bin in file 1
#! /pkg/bin/bash
[ssdavis@lect1 private]$ grepdir.sh . bin -l
```

```
./grepdir.sh
./cpdirs.sh
./makemake.sh
./temp2/1
./uncomp.sh
[ssdavis@lect1 private]$ grepdir.sh . BIN -l
[ssdavis@lect1 private]$ grepdir.sh . BIN -li
./grepdir.sh
./cpdirs.sh
./makemake.sh
./temp2/1
./uncomp.sh
[ssdavis@lect1 private]$ grepdir.sh . BIN li
usage: grepdir.sh directory pattern [-grep option]*
[ssdavis@lect1 private]$ grepdir.sh . BIN -n -i
1:#! /bin/bash
1:#! /bin/bash
1:#! /bin/bash
1:bin in file 1
1:#! /pkg/bin/bash
[ssdavis@lect1 private]$
```

2. (8 points) Write a shell script that copies the files in two directories dir1 and dir2 (supplied as the first two arguments), to dir3 (supplied as the third argument). If dir1 and dir2 contain the files with the same name, then the newer file should be copied to dir3. This does NOT look at the subdirectories of dir1 and dir2. Dir1 and dir2 must be existing directories, and dir3 may not be an existing regular file. The script should produce a usage statement if the script is misused. Name your script cpdirs.sh

```
[ssdavis@lect1 private]$ ls -lR
.:
total 24
-rwx------ 1 davis users  533 2010-01-10 14:59 cpdirs.sh
-rwx------ 1 davis users  395 2010-01-10 14:39 grepdir.sh
-rwxr--r-- 1 davis users  980 2010-01-10 12:19 makemake.sh
drwxr-xr-x 2 davis users 4096 2010-01-10 15:01 temp
drwxr-xr-x 2 davis users 4096 2010-01-10 15:04 temp2
-rwx------ 1 davis users  384 2010-01-10 12:17 uncomp.sh

./temp:
total 0
-rw-r--r-- 1 davis users 0 2010-01-10 15:06 1
-rw-r--r-- 1 davis users 0 2010-01-10 14:44 2
-rw-r--r-- 1 davis users 0 2010-01-10 14:44 3.c

./temp2:
total 0
-rw-r--r-- 1 davis users 15 2010-01-10 15:04 1
-rw-r--r-- 1 davis users  4 2010-01-10 15:02 2
-rw-r--r-- 1 davis users  0 2010-01-10 14:44 3.cpp
-rw-r--r-- 1 davis users  4 2010-01-10 15:02 4
[ssdavis@lect1 private]$ cpdirs.sh temp temp2 temp3
[ssdavis@lect1 private]$ ls -l temp3
total 0
-rw-r--r-- 1 davis users 0 2010-01-10 15:08 1
-rw-r--r-- 1 davis users 4 2010-01-10 15:08 2
-rw-r--r-- 1 davis users 0 2010-01-10 15:08 3.c
-rw-r--r-- 1 davis users 0 2010-01-10 15:08 3.cpp
-rw-r--r-- 1 davis users 4 2010-01-10 15:08 4
[ssdavis@lect1 private]$ cpdirs.sh temp temp2 temp3
[ssdavis@lect1 private]$ ls -l temp3
```

```
total 0
-rw-r--r-- 1 davis users 0 2010-01-10 15:09 1
-rw-r--r-- 1 davis users 4 2010-01-10 15:09 2
-rw-r--r-- 1 davis users 0 2010-01-10 15:09 3.c
-rw-r--r-- 1 davis users 0 2010-01-10 15:09 3.cpp
-rw-r--r-- 1 davis users 4 2010-01-10 15:09 4
[ssdavis@lect1 private]$ cpdirs.sh temp
usage: cpdirs.sh source_directory1 source_directory2 dest_directory
[ssdavis@lect1 private]$ cpdirs.sh temp makemake.sh
usage: cpdirs.sh source_directory1 source_directory2 dest_directory
[ssdavis@lect1 private]$ cpdirs.sh temp temp2 makemake.sh
usage: cpdirs.sh source_directory1 source_directory2 dest_directory
[ssdavis@lect1 private]$
```

3. (14 points) Write a shell script called uncomp.sh that will take a file with the standard archiving and compression extensions (gz, zip, tgz, and tar) and perform the appropriate operation[s] to uncomp.shress and unarchive the files. Acceptable filename extension combinations are .tar, .tar.gz, .tgz, .gz, and .zip. If the filename doesn't have one of these extensions then your script should print a usage statement. An example of this script is provided below. The usage is "uncomp.sh {filelist}+"

```
% tar -czf test.tar.gz temp
% ls
temp  test.tar.gz  uncomp.sh  upleast
% rm -rf temp
% ls
test.tar.gz  uncomp.sh  upleast
% uncomp.sh test.tar.gz
% ls
temp  test.tar.gz  uncomp.sh  upleast
% rm -rf temp
% tar -cf test3.pt.tar temp
% ls
temp  test.tar.gz  test2.tar.Z  test3.pt.tar  uncomp.sh  upleast
% rm -rf temp
% uncomp.sh test3.pt.tar
% ls
temp  test.tar.gz  test2.tar.Z  test3.pt.tar  uncomp.sh  upleast
% mkdir temp2
% zip test4.zip temp2
% ls
temp  temp2  test.tar.gz  test2.tar.Z  test3.pt.tar  test4.zip  uncomp.sh  upleast
% rm -rf temp*
% uncomp.sh test4.zip test3.pt.tar
% ls
temp  temp2  test.tar.gz  test2.tar.Z  test3.pt.tar  test4.zip  uncomp.sh  upleast
% uncomp.sh upleast
uncomp.sh: upleast has no compression extension.
% uncomp.sh
usage: uncomp.sh {filelist}+
% zip test.zip test*
  adding: test10.c (stored 0%)
  adding: test11.c (deflated 16%)
  adding: test1.c (deflated 18%)
  adding: test1.noc (deflated 18%)
  adding: test2.c (stored 0%)
  adding: test3.c (deflated 8%)
  adding: test4.c (deflated 8%)
  adding: test5.c (stored 0%)
  adding: test6.c (stored 0%)
  adding: test7.c (stored 0%)
  adding: test8.c (stored 0%)
  adding: test9.c (stored 0%)
% mkdir temp
% cp test.zip uncomp.sh temp
% cd temp
% ls
test.zip  uncomp.sh
```

```
% uncomp.sh test.zip
% ls
test10.c  test1.c    test2.c  test4.c  test6.c  test8.c  test.zip
test11.c  test1.noc  test3.c  test5.c  test7.c  test9.c  uncomp.sh
%
```

5. (21 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 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 p2]$ makemake.sh
Executable name required.
usage: makemake.sh executable_name
[ssdavis@lect1 p2]$ makemake.sh cal.out
[ssdavis@lect1 p2]$ 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 p2]$ make clean
rm -f cal.out appointment.o  calendar.o  day.o  dayofweek.o  time.o  year.o
[ssdavis@lect1 p2]$ makemake.sh cal.out -O2 -g
[ssdavis@lect1 p2]$ 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 p2]$
```