ECS 40 **Program #2 (46 pts)** **Spring 2015**

Due: Wednesday, April 15th, 11:59pm
Use handin to submit main.c, prime.c, cpdirs.sh, grepdir.sh, makemake.sh, and authors.txt to the p2 directory in cs40a.

gdb Tutorial (10 points)

The interactive gdb tutorial is available online from the course website: http://csiflabs.cs.ucdavis.edu/~ssdavis/40/gdb_Tutorial.pdf . You should submit main.c, prime.c, and your typescript from this tutorial. Each person must do the tutorial individually. You will find main.c, and prime.c in ~ssdavis/40/p2.

Bash Shell Scripts (36 points)

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.

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

```
[davis@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
[davis@lect1 private]$ cpdirs.sh temp temp2 temp3
[davis@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
[davis@lect1 private]$ cpdirs.sh temp temp2 temp3
[davis@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
[davis@lect1 private]$ cpdirs.sh temp
usage: cpdirs.sh source_directory1 source_directory2 dest_directory
[davis@lect1 private]$ cpdirs.sh temp makemake.sh
usage: cpdirs.sh source_directory1 source_directory2 dest_directory
[davis@lect1 private]$ cpdirs.sh temp temp2 makemake.sh
usage: cpdirs.sh source_directory1 source_directory2 dest_directory
[davis@lect1 private]$
```

3. (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. Do not use "*.o" in your clean routine. (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.)

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