**Regis University CC&IS**

**CS465 Unix**

**Lab Homework #5**

Complete the following nine exercises on a Unix system. Fill in your answers on this document. Points that will be given for each correct answer are listed by each question.

WARNING: The last script may take a while to write.

NOTE: You will have 1 ½ weeks to complete this homework.   
See the end of the document for submission instructions.

**Matt Hartigan**

**15-Apr-2019**

**\*answers below in red\***

### Exercises:

1. (8 points total – 2 points per answer) Use **cron** to send a mail message to yourself every Wednesday morning at 9 am reminding you to water your office plants.   
Complete the following tasks to do so:

a. Use **vi** to create a **crontab** file named **cronremind** that will send the reminder.

Place the contents of this crontab file here:

Contents of cronremind:

0 9 \* \* 3 mailx -s “Reminder: Water your office plants!” harti149

b. Register the crontab file.

What command did you use?

crontab cronremind

c. Use crontab to list the contents of the registered file.

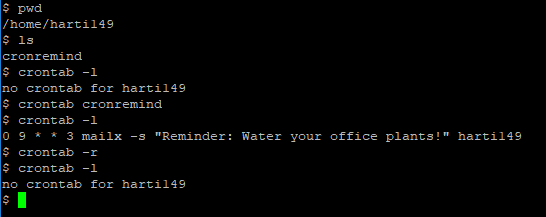
What command did you use?

crontab -l

d. Unregister the crontab file.

What command did you use?

crontab -r



2. (10 points total, allocation noted below) Use a shell script ***and*** the **at** command to do the same thing (send a mail message to yourself every Wednesday morning at 9 am reminding you to water your office plants). Complete the following tasks:

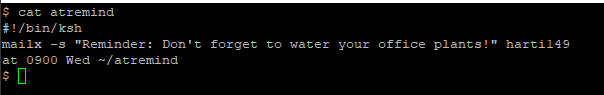
a. (4 points) Use **vi** to create script named **atremind** to send the reminder and then reschedule itself.  
  
Place the contents of this script file here:

Contents of atremind:

#!/bin/ksh

mailx -s "Reminder: Don't forget to water your office plants!" harti149

at 0900 Wed ~/atremind



b. (2 points) Use the **at** command to start the script 5 minutes from now.

What command did you use?

at now + 5 minutes < atremind

c. (2 points) Display a list of your scripts that are queued up to be run by the at command.

What command did you use?

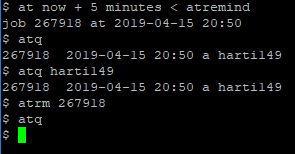
atq harti149

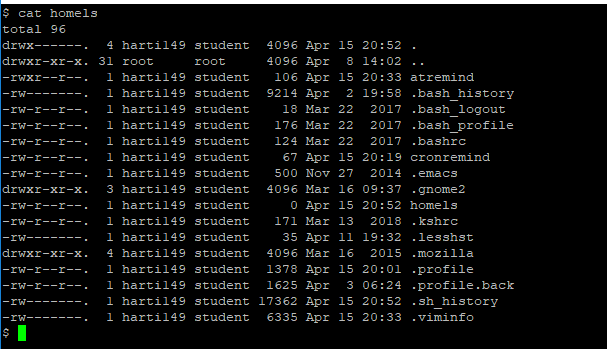
d. (2 points) Remove your script from the at queue.

What command did you use?

atrm 267918

(where 267918 is the job number in the queue)

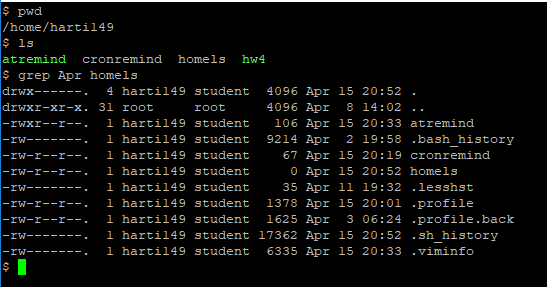
  
  
  
3. (10 points total – 2 points per answer) Create a long listing of your home directory, including hidden files, using **ls –al**, and save it to the file **homels**.



Then use the **grep** command to do the following tasks (submit all commands used):

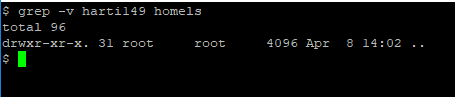
1. Display the ***lines*** in **homels** for files that were last accessed in the current month.

grep Apr homels



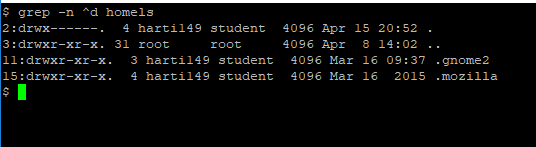
1. Display the ***lines*** in **homels** for the files that are **not** owned by you.

grep -v harti149 homels



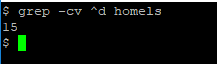
1. Display both the ***lines*** and ***line numbers*** of the lines in **homels** for the files that are directory files (i.e. the line starts with a "d").

grep -n ^d homels



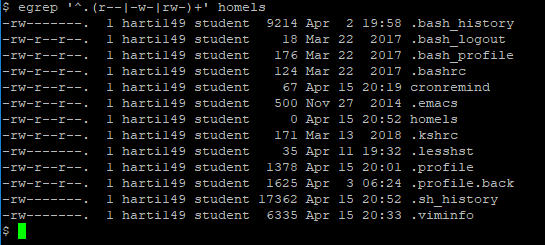
1. Display a ***count*** of the regular (non-directory) files in **homels**.

grep -cv ^d homels



1. Display all ***lines*** in **homels** where the **owner** has either read OR write permission OR both, but NOT execute permission on the file – any other permission settings, set or not, do not matter (HINT: Use **egrep** with a regular expression).

egrep '^.(r--|-w-|rw-)+' homels



4. (4 points) Write a shell script called **hw4,** whichwill use **grep** and any other Unix commands necessary to display only a **count** of how many users currently logged in AND are using the Korn shell.

Place the contents of the script file here:

hw4 script:

#!/bin/ksh

cat /etc/passwd | egrep 'usr/bin/ksh$' > hw4.txt

count=0

for item in `who -q`

do

if [ `egrep -c "^$item" hw4.txt` -eq 1 ]

then

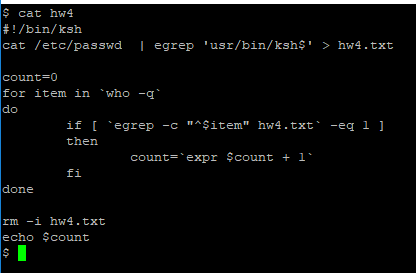
count=`expr $count + 1`

fi

done

rm hw4.txt

echo $count



5. (2 points) Write a command to create an alias called **kornusers** to run the **hw4** script.

alias -x kornusers=’~/hw4’

6. (2 points) Use **grep** (or **egrep**) commands to determine which files in your current directory have contents that contain EITHER the word "**system**" OR the word "**log**" (or both), and will display JUST the filenames of these files.

egrep -l ‘(system|log)+’ \*

7. (6 points) Write a Korn shell script called **hw7**, which will:

Use **grep** commands, along with any other necessary commands, to determine which files in your current directory have ***contents*** that contain BOTH the word "**system**" AND the word "**log**" (i.e. do not include files that contain only "system" or only "log").

The script should then concatonate the ***contents*** of those files found together into a new file called **syslogs**.

***HINT:*** First find the names of the files whose contents contain "system", then search THOSE files to find the files that also contain "log". Finally, concatonate the contents of the files found.

***WARNING:*** Although this script only requires a few lines, it may take some thought to figure it out.

NOTE: To test this script, you will need to create several files in your directory that contain the words "system" and "log", as well as some that contain only "system" or only "log". Only the files that contain **both** words should be concatonated into the syslogs file.

Place the contents of the script file here:

#!/bin/ksh

system=`grep -l 'system' \*`

log=`grep -l 'log' \*`

touch syslogs

for sysfile in $system

do

if [ -f $sysfile ]

then

for logfile in $log

do

if [ $sysfile = $logfile ]

then

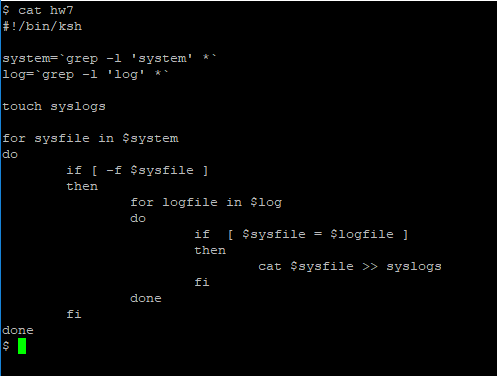
cat $sysfile >> syslogs

fi

done

fi

done



8. (28 points total, allocation noted below)

Write a Korn shell script called **hw8**, which will utilize the **select** command to create a menu of choices for the user.

The five actions will be the following:

- (1 points) Clear your screen  
  
- (4 points) Use the **find** command to find all files within your entire personal directory tree that end in **.tmp** and delete them from within the **find** command

BEWARE: If this choice is implemented incorrectly, you could wipe out ALL the files in your account! It would be a good idea to use the **–i** option with **rm**.

- (8 points) Use the **find** command to find all files within your entire personal directory tree that have not been modified in over 2 weeks, and move them to the subdirectory **$HOME/oldfiles** from within the **find** command (if this option is chosen by the user, your script should verify that subdirectory **oldfiles** exists BEFORE you try to move the files. If it does not exist, the script should create it.)  
  
- (9 points) Display a long listing of a subdirectory (If this option is chosen by the user, your script should prompt for a subdirectory name in a loop, until a valid subdirectory name is given)  
  
- (1 point) Exit the script

(1 point) Issue an error message if a valid option is not chosen.

(4 points) Let the user choose which action to perform, do the action, and then loop back to the menu until exit is chosen.

Place the contents of the script file here:

#!/bin/ksh

select choice in "Clear screen" "Delete all .tmp" "Sort old files" "Display subdir" "Exit"

do

case $choice in

"Clear screen")

clear;;

"Delete all .tmp")

echo "Delete all .tmp"

find $HOME -name '\*.tmp' -exec rm -i {} \;;;

"Sort old files")

olddir="$HOME/oldfiles"

if [ ! -d $olddir ]

then

echo "Creating directory for old files."

mkdir $olddir

fi

echo "Sorting old files."

find $HOME -mtime +14 -exec mv -t "$olddir" {} +;;

"Display subdir")

subdir=""

while [ ! -d "$subdir" ]

do

echo "Please enter a valid subdirectory name:"

read subdir

if [ -d $subdir ]

then

ls -l $subdir

fi

done;;

"Exit")

echo "Exiting script now."

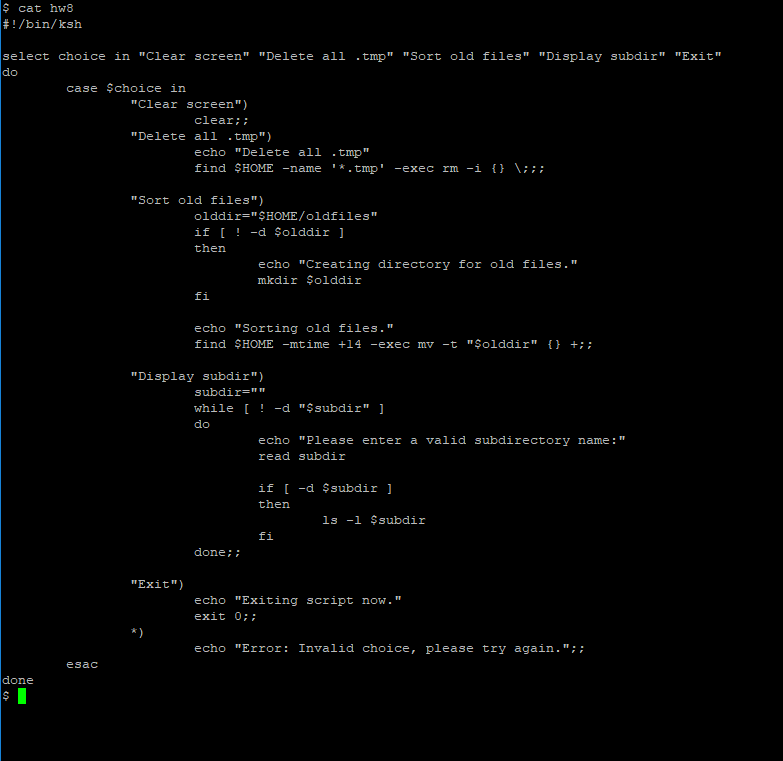
exit 0;;

\*)

echo "Error: Invalid choice, please try again.";;

esac

done



9. (30 points, allocated as noted below) Write a Korn shell script called **hw9**, which expands upon the script you wrote for **hw3** last week.

The script will still take three **command line arguments** a 3-letter month string (eg. Sep), a day number (eg. 18), and a 4-digit year (eg. 1996). No dates before Jan 1, 1920 or after the current year should be accepted.

*NOTE: Again, the current year should be accessed via the system* ***date****, not hardcoded into the script. The* ***date*** *command should not be used anywhere else in this script.*

(4 points) Validate all arguments (as in the week 4 homework). *Note that the* ***start year*** *is different from the week 4 homework.*

After argument validation, the script will calculate and display the number of days from Jan 1, 1920 to the date given in the command line.

Use the **let** command (( )) for all math calculations.

HINTS:

(14 points) First calculate all days that have passed in all years from 1920 to the year before the year entered.

(12 points) Then calculate the number of days that have passed in the year given, up to the date entered. ***Do not include the end date in your calculations*** (see sample output).

All leap years (i.e. extra leap days) in the date range should be taken into account. Note that for dates in this range, you will still only need to check if a year is evenly divisible by 4, to determine if it is a leap year.

NOTE: You will no longer need to convert the month string to a month number.

Sample outputs:

**$ hw9 Jan 5 1920**

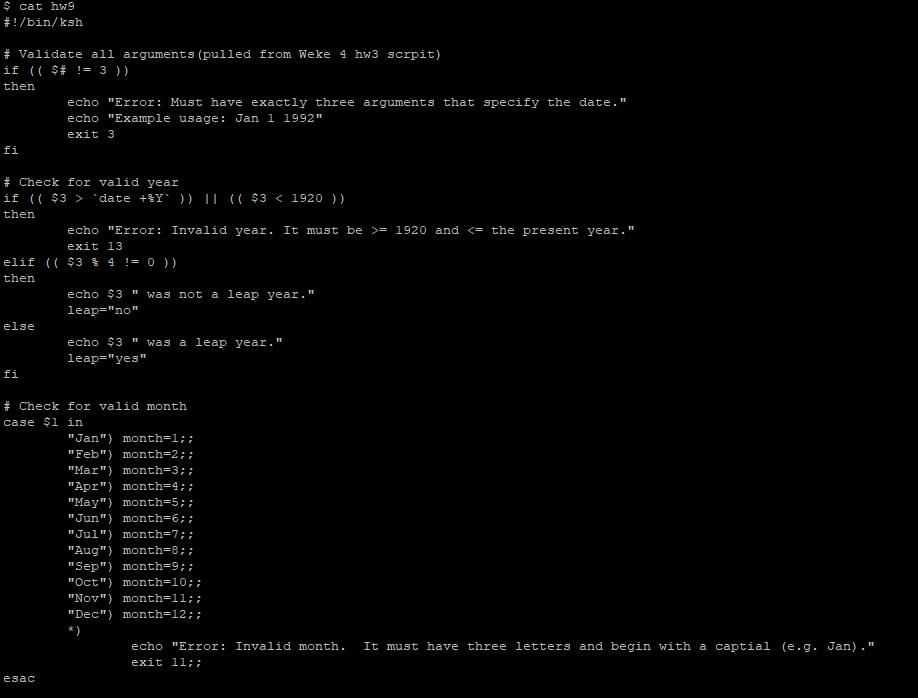
**4 days passed since Jan 1 1920**

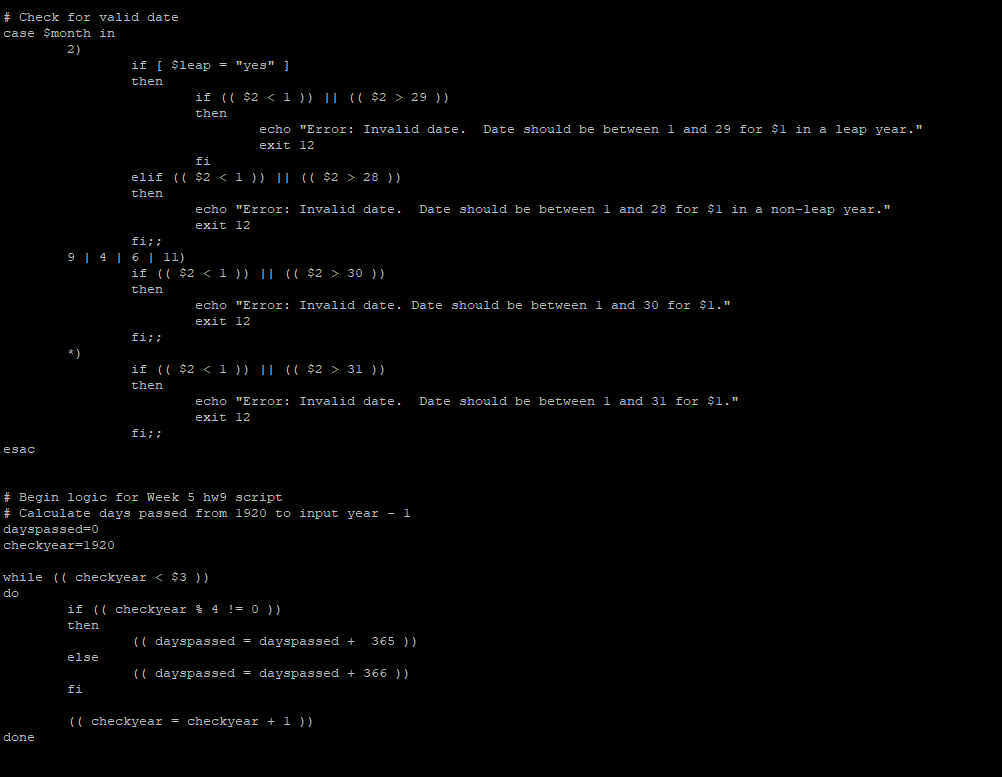
**$ hw9 Sep 20 1960**

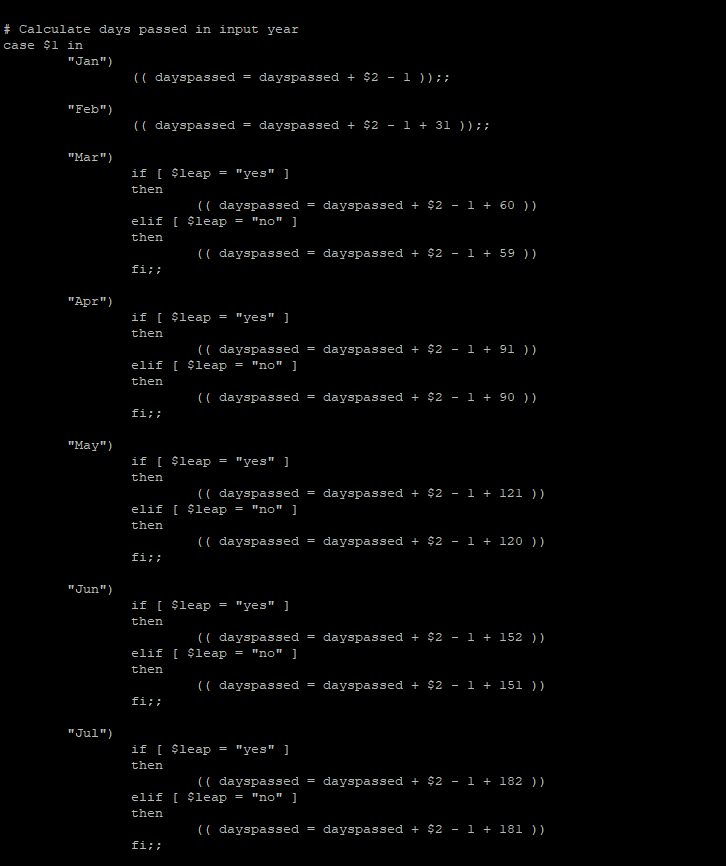
**14873 days passed since Jan 1 1920**

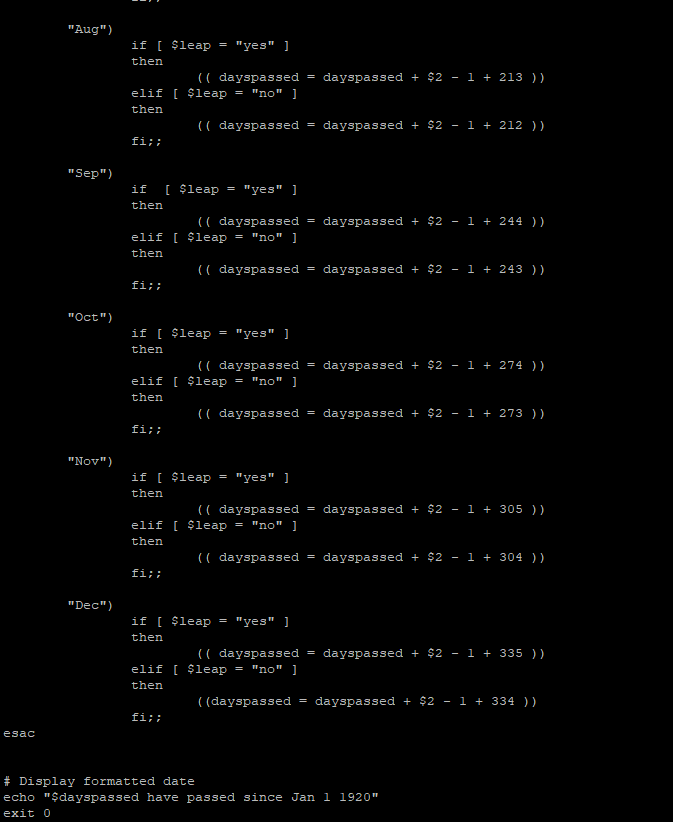
You can use the following website for error checking your calculations:

<http://www.timeanddate.com/date/duration.html>









WARNING:

This script requires quite a bit of programming logic, using various script control structures, and will be worth quite a few points. You will need to use at least one **case** statement, and many **if** statements.

Place the contents of the script file here:

#!/bin/ksh

# Validate all arguments(pulled from Weke 4 hw3 scrpit)

if (( $# != 3 ))

then

echo "Error: Must have exactly three arguments that specify the date."

echo "Example usage: Jan 1 1992"

exit 3

fi

# Check for valid year

if (( $3 > `date +%Y` )) || (( $3 < 1920 ))

then

echo "Error: Invalid year. It must be >= 1920 and <= the present year."

exit 13

elif (( $3 % 4 != 0 ))

then

echo $3 " was not a leap year."

leap="no"

else

echo $3 " was a leap year."

leap="yes"

fi

# Check for valid month

case $1 in

"Jan") month=1;;

"Feb") month=2;;

"Mar") month=3;;

"Apr") month=4;;

"May") month=5;;

"Jun") month=6;;

"Jul") month=7;;

"Aug") month=8;;

"Sep") month=9;;

"Oct") month=10;;

"Nov") month=11;;

"Dec") month=12;;

\*)

echo "Error: Invalid month. It must have three letters and begin with a captial (e.g. Jan)."

exit 11;;

esac

# Check for valid date

case $month in

2)

if [ $leap = "yes" ]

then

if (( $2 < 1 )) || (( $2 > 29 ))

then

echo "Error: Invalid date. Date should be between 1 and 29 for $1 in a leap year."

exit 12

fi

elif (( $2 < 1 )) || (( $2 > 28 ))

then

echo "Error: Invalid date. Date should be between 1 and 28 for $1 in a non-leap year."

exit 12

fi;;

9 | 4 | 6 | 11)

if (( $2 < 1 )) || (( $2 > 30 ))

then

echo "Error: Invalid date. Date should be between 1 and 30 for $1."

exit 12

fi;;

\*)

if (( $2 < 1 )) || (( $2 > 31 ))

then

echo "Error: Invalid date. Date should be between 1 and 31 for $1."

exit 12

fi;;

esac

# Begin logic for Week 5 hw9 script

# Calculate days passed from 1920 to input year - 1

dayspassed=0

checkyear=1920

while (( checkyear < $3 ))

do

if (( checkyear % 4 != 0 ))

then

(( dayspassed = dayspassed + 365 ))

else

(( dayspassed = dayspassed + 366 ))

fi

(( checkyear = checkyear + 1 ))

done

# Calculate days passed in input year

case $1 in

"Jan")

(( dayspassed = dayspassed + $2 - 1 ));;

"Feb")

(( dayspassed = dayspassed + $2 - 1 + 31 ));;

"Mar")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 60 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 59 ))

fi;;

"Apr")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 91 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 90 ))

fi;;

"May")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 121 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 120 ))

fi;;

"Jun")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 152 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 151 ))

fi;;

"Jul")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 182 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 181 ))

fi;;

"Aug")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 213 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 212 ))

fi;;

"Sep")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 244 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 243 ))

fi;;

"Oct")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 274 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 273 ))

fi;;

"Nov")

if [ $leap = "yes" ]

then

(( dayspassed = days:qpassed + $2 - 1 + 305 ))

elif [ $leap = "no" ]

then

(( dayspassed = dayspassed + $2 - 1 + 304 ))

fi;;

"Dec")

if [ $leap = "yes" ]

then

(( dayspassed = dayspassed + $2 - 1 + 335 ))

elif [ $leap = "no" ]

then

((dayspassed = dayspassed + $2 - 1 + 334 ))

fi;;

esac

# Display formatted date

echo "$dayspassed have passed since Jan 1 1920"

exit 0

# Submission

This homework assignment is due by midnight Wednesday, next week (Week 6).

Submit a filled in copy of this Word document to the **Homework Assn 5** drop box (located under the Dropbox tab in the online course).

Before submitting the Word file with your answers, you MUST rename it as follows:

### Lastname-hwk5.docx

For example:

### Smith-hwk5.docx