**CS 3421 Week 3 Problems: Shell Variables, Jobs, Shell Scripts**

**1. Shell Variables**

1a: Create a shell script, whereAmI, which prints: “Your current directory is ‘/your/dir’, your home directory is ‘/home/yourdir’” The paths should correspond to your actual directories.

Vim whereAmI.bash

echo “Your current directory is $PWD, your home directory is $HOME”

1b: Create a shell script, twiceParams, which prints the total number of parameters multiplied by two.

Vim twiceParams

echo $(( $# \* 2 ))

1c: Create two scripts, test1 and test2. Script test1 should create two variables, one=1 and two=2, run the test2 script, and then print $one and $two. Script test2 should *only* contain the following:

#test2

one=one

two=two

Contents of test1?

one=1

two=2

./test2.bash

echo $one $two

Modify test1 in such a way that, when run, it prints the assignments from test2 instead of test1. Do not modify the variable values in either script.

one=1

two=2

source test2.bash

echo $one $two

**2. Jobs**

2a: Create a job sequence which changes your current directory to ~/courses/cs3421 and, if successful, lists the directory’s contents.

cd ~/courses/cs3421 && ls

2b: Create a job sequence which changes your current directory to ~/courses/cs3421 and, if it doesn’t exist, creates the directory.

cd ~/courses/cs3421 || mkdir ~/courses/cs3421

2c: Create a job sequence which changes your current directory to ~/courses/cs3421 and, if successful, lists the directory’s contents. If it is not successful, it should create the directory.

Cd ~/courses/cs3421 && ls || mkdir ~/courses/cs3421

**3. Shell Scripts**

3a: Create a shell script, multiplier, which prompts the user, “Enter two numbers to multiply:”, reads two values in from stdin (not as parameters), and prints the product of the two. Do not worry about validation (just be sure to only enter numbers).

read –p “Enter two numbers to multiply:” number\_one number\_two

echo $(( $number\_one \* $number\_two ))

3b: Create a shell script, multTwo, which takes two parameters and prints their product. Create the script so that if exactly two arguments aren’t supplied, it will print the error, “usage: multTwo [number1] [number2]”. If you change the name of the script (e.g., from “multTwo” to “multTwoParams”), the script should automatically print the correct name in the error message.

if [ $# -ne 2 ]  
then  
 echo “usage: $0 [number1] [number2]”  
 exit 1  
fi  
echo $(( $1 \* $2 ))

3c: Create a shell script, mathLines, which can take a file’s contents from stdin containing an arbitrary number of integers separated by newlines and prints their product and sum. Note: be sure that you are adding the numbers and not concatenating them.

sum=0  
product=1  
  
while read line  
do  
 sum=$(( $sum + $line))  
 product=$(( $product \* $line))  
done  
  
echo “Sum is $sum”  
echo “Product is $product”  
  
called by ./mathLines.bash < filename.ext

3b: Create a shell script, findFirst, which reads a file’s contents line by line and prints the line number of the *first occurrence* of the line reading “findme”.

linenum=0  
  
while read line  
do  
 linenum=$(( $linenum +1 ))  
 if [ $line = “findme” ]  
 then  
 echo “Found on line: $linenum”  
 exit 0  
 fi  
done  
  
echo “Not found”