ULI101- Final test review

Part A - Matching (can have multiple matches)

1. Symbol for the parent directory
2. Symbol that represents home directory
3. Display contents of a directory and option to view in detail.
4. Create a file
5. Changing directories
6. Removing files recursively
7. Create directories and its subdirectories respectively
8. Copying files or directories recursively
9. To view detailed information about a certain command
10. Check working directory
11. Quickly return to your home directory
12. Modify permissions
13. To move/ rename a file/ directory
14. Symbol for root directory
15. The command to count lines
16. The command to extract parts of a file
17. The command to display last 10 lines of a file
18. The command to display the first 10 lines of a file
19. Search for certain pattern from a file
20. The command to create a soft/ Symbolic link

a)	mkdir - p
b)	man
c)	$
d)	sort
e)	cp
f)	chmod
g)	/
h)	~
i)	wc
j)	tail
k)	link
l)	ls -l
m)	sort
n)	ls
o)	grep
p)	rm -r
q)	mv
r)	wc -w
s)	cp - r
t)	\
u)	cd
v)	ln -s
w)	pwd
x)	cut
y)	touch
z)	wc -l
aa)	ln
bb)	head
cc)	move
dd)	..

ULI101- Final test review

Part B - Unix Commands

For Question 1 to 7 use the file below.
Note: The file format is as follows: First name, Last name, Age, ID
Assume all questions are independent of one another and the file is in your current directory.
Your answer must be in a single line

```
$> cat contacts.csv

Rick,Sanchez,70,23456
Morty,Smith,14,23224
Beth,Smith,34,54221
Jerry,Smith,35,45632
Summer,Smith,17,37463
Gene,Vince,71,47362
???,Mr.Meeseeks,???,45324
Jessica,???,16,74638
Doofus,Rick,70,98765
Evil,Morty,14,76453
Scary,Terry,???,99999
Noob,Noob,25,11001
```

1. Format the file so that each field is separated by a space instead of comma and sorted by last name and store the output to a file called "contactsmod.txt" (Hint: Use `tr` command)

2. Sort the file and display the output that do not end in a number 1 to 5.

3. Sort the file by in descending format and store the lines 6 to 10 in a file called "sortedDesc.txt" and display it on the screen.

4. Using SED replace all the invalid first names i.e. "???" with the name of your choice and save it into a new file called "validFirstnames.txt"

5. Grab all the lines with age below 70, store the output into a file called "ageLT70.csv" and display the total lines stored into the file. (Hint: Don't use grep/sort/cut)

ULI101- Final test review

Part B - Unix Commands (continue)

6. Replace all the instances of "smith" or "Smith" to "SMITH" in the same file. (Hint: Don't use tr or grep)

7. Write a command to print all the VALID records (i.e. records without "???") of file contacts.csv in the following format "Firstname Lastname is age years old" and store the result in the same file.

For Question 8 to 11 use the file (cars) below.
Note: The file format is as follows: Car make, Model, Year, Price
Assume all questions are independent of one another and the file is in your current directory.
Your answer must be in a single line

```
$> cat cars

    Nissan,300ZX,1993,2305
    Mitsubishi,Pajero,1992,
    Saturn,Ion,2003,11760
    Pontiac,Tempest,1961,11500
    Bugatti,Veyron,2009,1700000
    Toyota,T100 Xtra,1995,2595
    Mitsubishi,Mirage,1996,1618
    Ford,Econoline E250,2000,1285
    Maserati,Spyder,2005,87252
```

8. `head -5 cars | awk -F, '$4 > 3152 {print $1 " " $2 " costs $" $4}' | sort`

9. `awk -F, '$3 >= 2000 && $3 < 2009 {print "The " $1 " " $2 " is of the year " $3}' cars`

ULI101 - Final test review

Part B - Unix Commands (continue)

10. `grep -i ",.....$" cars| sed 's/,/ /g' | sort -nrk4 | cut -d" " -f1,2 2>&1 >sortedExpCars.txt`

SCREEN	SORTEDEXPCARS.TXT

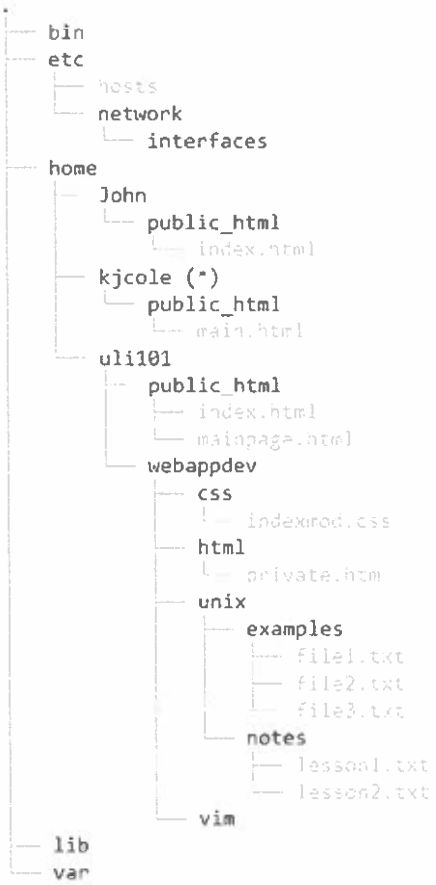
11. `sed '1 s/[0-9]/*/' cars | head -1`

12. `egrep "([0-9 | a-zA-Z])*\ ([EXa-t | 250])*" cars`

13. `sort cars | tail -5 | head -2`

ULI101- Final test review

Part C - File management



All answers are based on the following tree diagram displayed below. Assume all questions are independent of one another. `john` and `uli101` belong to <u>different</u> groups. Write all the answers in single line.

Your user id: `kjcole`,
Your PWD: `/home/kjcole`

1. Use an Absolute path to change the name of the directory `examples` to `docs`

2. Identify the type of paths (Relative, Relative to home or Absolute) for the following.

   a. `cp ~/public_html/* .`

   b. `mv /home/uli101/notes/lesson2.txt ../`

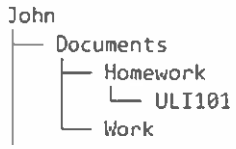
   c. `ls -l ..`

   d. `touch ~John/a.cpp ./b.cpp /c.cpp`

ULI101- Final test review

Part C - File management (continue)

3. Write a single Unix command to add the following directory path starting from `John/`



4. Write a single Unix command to give `John` and `ULI101`, read and write permissions on your `public_html`

5. Write a command in SINGLE line without using ';' to create a directory under `John` called "`notes`" and copy everything from `notes` under user `uli101` to the `notes` you created.

6. Write a command to remove all the folder examples under the user uli101.

Absolute Path:

Relative to home:

Relative Path:

ULI101- Final test review

Part E– Scripting

1. Write the output of the script called `grades` below based on following calls. You can assume that script below is called grades and is in your current directory with appropriate permissions. Write the output/error based on the question.

```
#!/bin/bash

read -p "Please Enter a Grade Number: " grade

if [[ $grade -ge 90 && $grade -le 100 ]];then
        echo "congratulations, you received an A+"
elif [[ $grade -ge 80 && $grade -le 89 ]];then
        echo "Very Good! you got an A"
elif [[ $grade -ge 70 && $grade -le 79 ]]; then
        echo "Good! you got a B!"
elif [[ $grade -ge 60 && $grade -le 69 ]]; then
        echo "you need to work a bit more but not bad! a C!!"
elif [[ $grade -ge 50 && $grade -le 59 ]]; then
        echo "ouch.. you need to stop playing games! you got a D!!"
elif [[ $grade -gt 0 && $grade -le 50 ]]; then
        echo "Well.. It happens. You know."
else
        echo ".....I dont get what you are trying to tell me....."
        exit 1
fi
```

   a. ./grades

      If the script runs, enter the value 59 as an input and write the expected output below:

   b. ./grades 75

      If the script runs, enter the value 75 as an input and write the expected output below

ULI101- Final test review

Part E– Scripting (continue)

c. grades

If the script runs, enter the value 90 as an input and write the expected output below:

2. Write a script which fulfills the following requirements. This script allows user to create files. Minimum files will be at least 5. You can assume that user will only enter numbers as an argument while calling the script. (Optional)

   a. Name of the program/script should be `createFiles`
   b. The program must check if there is a positional parameter and that parameter must have a number greater than 5. In the events of missing parameter, the program should handle the situation and print the following line as an error "`USAGE: createFiles num-of-Files`". However, if there is a parameter. The program should check the value and see if it fulfills the requirement of minimum 5 files. If it does not, it writes 5 files by default and prints the following message: "Setting the default number to 5" and should write 5 files in the current directory.
   c. The name of file would follow the following pattern: filenum.txt. Example: file1.txt, file2.txt etc.
   d. The files should contain the following line inside them,
      "This filename was created on date"

Expected Output:

```
$ >createFiles
USAGE: createFiles Num-of-Files
$ >createFiles 4
Setting the default number to 5
$ >ls file?.txt
file1.txt  file2.txt  file3.txt  file4.txt  file5.txt
$ >cat file?.txt
This file1.txt was created on Mon Nov 20 12:54:17 STD 2017
This file2.txt was created on Mon Nov 20 12:54:17 STD 2017
This file3.txt was created on Mon Nov 20 12:54:17 STD 2017
This file4.txt was created on Mon Nov 20 12:54:17 STD 2017
This file5.txt was created on Mon Nov 20 12:54:17 STD 2017
```