Question 1

---

awk

- Description: Awk allows programmers to create simple scripts for file manipulation
- Formula:

```
awk instruction file
```

- Examples: Prints collumns 1 through 4

  ```
  awk '{print $1 "\t" $4}` collumns.txt
  ```

  Displays all lines of a length greater than 10

  ```
  awk 'length($0) > 10' lines.txt
  ```

  Prints all lines that have an e

  ```
  awk '/e/ {print $0}' file.txt
  ```

---

cat

- Description: Cat displays files. Cat is also used to concatenate files
- Formula:

  ```
  cat +option +file
  ```

- Examples:
  - Display the code of a program

  ```
   cat /myProjects/helloWorld.cpp
  ```

  - Displays a $ at the end of each line

```
cat -E file.txt
```

  - Displays tabs characters as ^I

```
cat -T file.txt
```

---

cp

- Description: Copies a file to a specified output
- Formula:

```
cp option source_file destination
```

- Examples: Copy a file from the home directory into the docs directory

```
cp user/home/file1.txt user/docs/file1.txt
```

Make a symbolic link instead of copying

```
cp -s file2.txt file2Link
```

Copy only if updating(eg. the source file is not as new or doesn't exist)

```
cp -u fileOrig.txt fileUp.txt
```

---

cut

- Description: Splices a certain section from a file
- Formula:

```
cut + option + file
```

- Examples: Cut using a custom delimiter from lines 1-8

```
cut -d "x" -f 1-8 book.txt
```

Grab the complement of the above command

```
cut -d "x" -f 1-8 book.txt
```

Cut by the @ character

```
cut -c @ record.txt
```

---

grep

- Description: Grep searches for patterns in a file
- Formula:

```
grep +options +pattern +file
```

- Examples:

---

head

- Description: Displays the first 10 lines of the file
- Formula:

```
head +option + files
```

- Examples: Displays the first line of file1

```
head -n 1- file1.txt
```

Displays

---

ls

- Description: Display the contents of a directory
- Formula:

```
ls + options + directory
```

- Examples: Displays the current directory

```
ls
```

Displays hidden files in the current directory

```
ls -a
```

Displays a human readable list

```
ls -h
```

---

man

- Description: Used to access manual entries
- Formula:

```
man +command
```

- Examples: Get the manual for ls

```
man ls
```

Search the manual for occurences of the string '-h'

```
man -k '-h' ls
```

Display debugging information for a given command

```
man -d command
```

---

mkdir

- Description: Used for making directories
- Formula:

```
mkdir +options +directory
```

- Examples: Make a Favorites Directory

```
mkdir favorites
```

Assuming the favorites directory doesn't exist, make the favorited directory and a child directory called "classics"

```
mkdir -p favorites/classics
```

Use bracket expansion to make multiple sub directories for favorites in one line

```
mkdir -p favorites/{classics, new, pending}
```

---

mv

- Description: Used to move a file fom one location to another
- Formula:

```
mv options source_file new_location
```

- Examples:

Move all text files to the text directory

```
mv *.txt ~/text
```

Get feedback on what is being done

```
mv -v file1 /Directory
```

Do not prompt before overwriting

```
mv -f newFile.txt /Directory
```

tac

- Description: Displays the file in reverse line order
- Formula:

```
tac +options +file
```

- Examples: Displays the hello world program in reverse

```
tac helloWorld.cpp
```

Print the first line of the program

```
tac helloWorld.cpp | tail -n 1
```

Print the file backwards with a custom separator

```
tac -s "%LINE5%" helloworld.cpp
```

---

tail

- Description: Displays the last 10 line of a file
- Formula:

```
tail +options +file
```

- Examples: Displays the very last line of a file

```
tail -n 1 file.txt
```

```

```

```

```

---

touch

- Description: Archaically it was used to touch files and leave a record that they had been modified. Now it is mostly used to create new files.
- Formula:

```
touch +option +file
```

- Examples:

```
touch newFile.txt
```

Only check, do not create

```
touch -c checkedFile.txt
```

Create multiple files using bracket expansion

```
touch Documents/{doc{1,2,3,4,5,6,7,8,9}.txt}
```

---

tr

- Description: A tool for translating or deleting characters
- Formula:

```
tr +option +set1 +set2
```

- Examples: Replace all instances of i in the input with o

```
echo typi | tr i o
```

Replace all digits in the original with o

```
echo typi | tr [:digit:] o
```

Replace all Punctuation marks with a question mark

```
echo Is this a question. | tr [:punct:] ?
```

---

tree

- Description: Displays the file directory as a tree
- Formula:

```
tree +options +directory
```

- Examples: Display the documents folder as a tree

```
tree documents
```

Displays the documents folder in human readable format

```
tree -h documents
```

Displays just the sub directories in documents

```
tree -d documents
```

---

vim/nano

- Description: Text editors available in the command line

- Formula: nano +options +line,column +file
- Examples: Enable mouse support for nano

```
nano -m
```

Set the operating directory to projects

```
nano -o ~/projects
```

Display line numbers

```
nano -l
```

#Question 2 #

- How to work with multiple terminals open? Depending on the distro, multiple shortcuts can be used to either open a new terminal window. or split the pre existing terminal. I personally prefer to split to keep the workspace clear of cutter.
- How to work with manual pages? Manual pages for a given command can be accessed with

  ```
  man command
  ```

- How to parse (search) for specific words in the manual page

  ```
  man command | grep -- specific word
  ```

- How to redirect output (> and |) | pipes the left term as input for the right term

  > outputs the left into a file

- How to append the output of a command to a - file Use the > command

  ```
  echo "g" file > g.txt
  ```

  will create a text file containing "g"
- How to use wildcards for copying and moving multiple files at the same time "*" can stand in for any characters and any amount of characters ? looks for one character [] looks for a given set of characters We can have a different use case for each command. mv *.txt will move all text files mv*

book-100-???.txt will move all 100 level books mv [1-9].txt will move all text files that start with a number

- How to use brace expansion
    - For creating entire directory structures - in a single command

```
mkdir -pmkdir -p
Recipes/{Barbecue/{Northern,Souuthern},Steaming,Cocktails/{Virgin,Esot
eric}}
```

This will create a directory called Recipes with 3 folders and 4 sub folders, distributed through two of the folders.