Final Assignment

awk

Description: Awk acts like a versatile assistant, helping you find and process specific information in text files.

```
Syntax: awk 'pattern { action }' filename

Examples:

awk '/pattern/ { print $1 }' data.txt

awk '{ total += $2 } END { print "Total: " total }' data.csv

awk '{ print $1, $3 }' records.txt
```

cat

Description: Cat it shows the content of files by stitching them together.

```
Syntax: cat [options] file1 file2 ...

Examples:

cat file1.txt file2.txt

cat document.txt

cat intro.txt chapter1.txt conclusion.txt > book.txt
```

cp

Description: Cp is to creating a backup it copies files or directories.

```
Syntax: cp [options] source destination

Examples:

cp file.txt backup/

cp -r source directory/ destination_directory/

cp file1.txt file2.txt file3.txt new directory/
```

cut

Description: Cut it removes sections from each line of a file.

```
Syntax: cut [options] filename

Examples:

cut -c1-5 file.txt

cut -d',' -f1 data.csv

cut -c3-8 textfile.txt
```

grep

Description: Grep is a searching for patterns in files.

```
Syntax: grep [options] pattern file

Examples:

grep "error" logfile.txt

grep "search term" document.txt

grep -r -o "keyword" /path/to/directory | wc -l
```

head

Description: Head is reading the beginning of a book it shows the first part of a file.

```
Syntax: head [options] filename

Examples:

head -n 5 file.txt

head file.txt

head -c 50 data.csv
```

ls

Description: Ls lists the contents of a directory.

```
Syntax: ls [options] [files]

Examples:

ls -l

ls -a

ls
```

man

Description: Man shows the help page for a command.

```
Syntax: man command

Examples:

man ls

man -l grep

man cp
```

mkdir

Description: Mkdir it creates directories.

```
Syntax: mkdir [options] directory name

Examples:

mkdir new directory

mkdir movies music books

mkdir -p home/Document/movies
```

mv

Description: Mv it moves or renames files and directories.

```
Syntax: mv [options] source destination
```

```
Examples:

mv file.txt destination/

mv book.txt bible.txt

mv *.txt target directory/
```

tac

Description: Concatenate and display the content of files in reverse.

```
Syntax: tac file1 file2

Examples:

tac document.txt

tac file1.txt file2.txt

tac intro.txt chapter1.txt conclusion.txt > book.txt
```

tail

Description: Display the last part of a file.

```
Syntax: tail options file

Examples:

tail file.txt

tail -n 20 file1.txt file2.txt

tail -f error log.txt
```

touch

Description: Create an empty file or update the access/modification timestamps.

```
Syntax: touch file1 file2

Examples:

touch new file.txt
touch file1.txt file2.txt

touch document1.txt document2.txt document3.txt
```

tr

Description: Translate or delete characters from a text stream.

```
Syntax: tr options set1 set2

Examples:

tr 'a-z' 'A-Z' < input.txt > output.txt

tr -d ' ' < text with spaces.txt > no spaces.txt

tr '\t' ' ' < input with tabs.txt > output with spaces.txt
```

tree

Description: Display the directory structure as a tree.

```
Syntax: tree options directory

 Examples:

 tree

 tree -h

 tree /path/to/directory
```

Question 2

How to work with multiple terminals open?

To work with multiple open terminals, you can use terminal multiplexers like tmux or screen to split the screen and run commands simultaneously. You can also take advantage of multiple tabs or windows in terminal emulators, such as GNOME Terminal or Windows Terminal.

How to work with manual pages?

To access manual pages in Unix like systems, simply use the man command followed by the command or topic you want more information about.

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

Parsing through a manual page is reminiscent of searching for keywords in a book. By utilizing the man command with -k or employing a pipe (|) with grep

How to redirect output (> and |)

Redirecting output is to guiding the outcome of a task to a designated location. Using > is comparable to saving the output to a file, resembling the act of preserving results in a specific document.

How to append the output of a command to a file

Appending output is like adding new insights to the end of an existing document. When you use >>, it's as if you're expanding on the content of a file, much like appending additional notes or updates to the conclusion of a report without overwriting what's already there.

How to use wildcards

Think of wildcards as your helpful assistants in the digital world they're like flexible placeholders that can stand in for a variety of possibilities.

How to use brace expansion

Brace expansion is akin to creating a set of interconnected items in one go. It's like establishing multiple folders or directories simultaneously