# Question 1

## awk

- Description:
    - awk is a command-line utility for processing text files. It is particularly useful for pattern scanning and processing. It takes an input file or standard input, and performs actions on selected text.
- Formula:
- `awk + options + {awk command} + file`
- `command output | awk + options + {awk command}`
- Examples:
    - How to print the first field of a file:
        - `awk -F':' '{print $1}' /etc/passwd`
    - Print lines that contain the word "apple":
        - `awk '/apple/ {print}' file.txt`
    - How to change a field to upper case:
        - `awk -F: '{print toupper($1)}`

## cat

- Description:
    - cat is a command that allows you to view the contents of one or more files in the terminal.
- Formula:
- `cat [options] file`
- Examples:
    - Display the contents of a file:
        - `cat file.txt`
    - Concatenate two files and display the result:
        - `cat file1.txt file2.txt`
    - Append the contents of a file to another file:
        - `cat file1.txt >> file2.txt`

## cp

- Description:
    - cp is a command that allows you to copy files or directories.

- Formula:
- `cp [options] source_file destination_file`
- Examples:
    - Copy a file to a new location:
        - `cp file.txt new_file.txt`
    - Copy a directory and all its contents to a new location:
        - `cp -r dir/ new_dir/`
    - Copy a file to a remote server using SSH: *`scp file.txt user@remote:/path/to/destination`

## cut

- Description:
    - cut is a command that allows you to extract sections from each line of a file.
- Formula
- `cut [options] file`
- Examples:
    - Extract the first three characters from each line of a file:
        - `cut -c 1-3 file.txt`
    - Extract the third field from a file delimited by commas:
        - `cut -d ',' -f 3 file.txt`
    - Extract the last field from a file delimited by spaces:
        - `cut -d ' ' -f 4 file.txt`

## grep

- Description:
    - grep is a command that searches for a pattern in a file or set of files. It is particularly useful for finding specific text within large files.
- Formula:
- `grep [options] pattern [file]`
- Examples:
    - Find lines that contain the word "apple":
        - `grep "apple" file.txt`
    - Find lines that do not contain the word "orange":
        - `grep -v "orange" file.txt` Find lines that contain the word "apple" or "orange":
        - `grep -E "apple|orange" file.txt`

## head

- Description:
    - head is a command that displays the first few lines of a file.
- Formula: *`head [options] file`
- Examples:
    - Display the first 10 lines of a file:
        - `head file.txt`
    - Display the first 20 lines of multiple files:
        - `head -n 20 file1.txt file2.txt`
    - Display the first 5 lines of a file and then exit:
        - `head -n 5 file.txt; exit`

## ls

- Description:
    - ls is a command that lists the files and directories in the current directory.
- Formulas
- `ls [options] [directory]`
- Examples:
    - List all files and directories in the current directory:
        - `ls`
    - List all files and directories in a specific directory:
        - `ls /path/to/directory`
    - List all files and directories, including hidden files:
        - `ls -a`

## man

- Description:
    - Man is a command-line tool used to display the manual pages for a command. It provides detailed information about how to use a command, including its options and syntax.

*Formulas:

- `man [command]`

*Example: * Displays the manual page for the ls command * `man ls` * Displays the manual page for the grep command * `man grep` * Displays the manual page for the man command * `man man`

## mkdir

- Description:

    - Mkdir is a command-line tool used to create a new directory. It can be used to create a new directory in the current directory or in a specified directory.

- Formula: `mkdir [options] directory`
- Example:
    - Creates a new directory named mydir in the current directory
        - `mkdir mydir`
    - Creates a new directory named mydir in the /home/user directory
        - `mkdir /home/user/mydir`
    - Creates a new directory named mydir2 in the /home/user/mydir1 directory, and creates the mydir1 directory if it does not exist
        - `mkdir -p /home/user/mydir1/mydir2`

## mv

- Description
    - Mv is a command-line tool used to move or rename files and directories. It can be used to move a file or directory to a different location or to rename a file or directory.
- Formula: `mv [options] source destination`
- Example:
    - Renames file1.txt to file2.txt
        - `mv file1.txt file2.txt`
    - Moves file.txt to the /home/user directory
        - `mv file.txt /home/user/`
    - Moves directory1 to the /home/user directory
        - `mv directory1 /home/user/`

## tac

- Description:
    - Tac is a command-line tool used to display the contents of a file in reverse order. It can be used to view the end of a file without having to scroll through the entire file.
- Formula: `tac [options] file`
- Example:
    - Displays the contents of the file in reverse order
        - `tac file.txt`
    - Displays the contents of the file in reverse order, with a space separator
        - `tac -s ' ' file.txt`
    - Displays the contents of the file in reverse order, with no separator
        - `tac -r file.txt`

## tail

- Description:
  - Tail is a command-line tool used to display the last few lines of a file. It can be used to view the end of a file without having to scroll through the entire file.
- Formula: `tail [options] file`
- Example:
  - Displays the last 10 lines of the file
    - `tail file.txt`
  - Displays the last 5 lines of the file
    - `tail -n 5 file.txt`
  - Displays the last 10 lines of the file and follows the file as it grows
    - `tail -f file.txt`

## touch

- Description:
  - Touch is a command-line tool used to create an empty file or update the modification time of an existing file. It can be used to create a new file or to update the timestamp of an existing file.
- Formula: `touch [options] file`
- Example:
  - reates a new file named file
    - `touch file.txt`
  - To update the modification time of an existing file named example.txt
    - `touch example.txt`
  - Create multiple files named "file1.txt", "file2.txt", and "file3.txt" in a single command
    - `touch file1.txt file2.txt file3.txt`

## tr

- Description:
  - Tr is a command-line tool used to translate or delete characters from a file or stream. It can be used to replace one character with another or to delete certain characters from a file or stream.
- Formula: tr `[options] set1 set2`
- Example:
  - Translates 'e' and 'l' characters in the word 'hello' to '1' and '2', respectively
    - `echo "hello" | tr 'el' '12'`
  - Deletes all whitespace characters from the file

        - `cat file.txt | tr -d '[:space:]`
    - Translates all lowercase characters in the word 'hello' to uppercase
        - `echo "hello" | tr '[:lower:]' '[:upper:]`

### tree

- Description
    - Tree is a command-line tool used to display the directory structure in a tree-like format. It can be used to visualize the contents of a directory and its subdirectories.
- Formula: `tree [options] [directory]`
- Example:
    - Displays the directory structure of the current directory
        - `tree`
    - Displays the directory structure of the /home/user directory
        - `tree /home/user`
    - Displays the directory structure of the /home/user directory, up to a depth of 2 levels
        - `tree -L 2 /home/user`

# Question 2

## How to work with multiple terminals open?

You can open multiple terminals on your operating system by launching the terminal application multiple times. Each terminal will have its own prompt and command line interface, allowing you to execute commands in separate instances.

## How to work with manual pages?

Manual pages are documentation pages that provide detailed information about commands and other system components. To access the manual page for a command, type "man [command]" in the terminal. For example, to view the manual page for the "ls" command, type "man ls". Once the manual page is displayed, you can use the arrow keys to navigate through it, press "q" to quit, or search for specific words by pressing the "/" key followed by the search term.

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

As mentioned above, you can search for specific words in a manual page by pressing the "/" key followed by the search term. This will highlight all occurrences of the search term in the manual page.

## How to redirect output (> and |)?

You can redirect the output of a command to a file by using the ">" symbol followed by the file name. For example, to redirect the output of the "ls" command to a file named "directory_contents.txt", type "ls > directory_contents.txt" in the terminal. You can also use the "|" symbol to pipe the output of one command

to another command. For example, to display only the first 10 lines of the output of the "ls" command, type "ls | head -10" in the terminal.

## How to append the output of a command to a file?

To append the output of a command to a file, you can use the ">>" symbol followed by the file name. For example, to append the output of the "ls" command to a file named "directory_contents.txt", type "ls >> directory_contents.txt" in the terminal.

## How to use wildcards for copying and moving multiple files at the same time?

Wildcards are special characters that can be used to represent multiple characters in a file name. For example, the "*" character represents any number of characters. To copy or move multiple files at the same time using wildcards, use the "cp" or "mv" command followed by the file names that match the desired pattern. For example, to copy all files in the current directory that have a ".txt" extension to a new directory called "text_files", type "cp *.txt text_files" in the terminal.

## How to use brace expansion for creating entire directory structures in a single command?

Brace expansion is a feature that allows you to generate multiple strings based on a pattern. To create entire directory structures in a single command using brace expansion, use the "mkdir" command followed by the desired directory structure pattern enclosed in braces. For example, to create a directory structure with three subdirectories called "a", "b", and "c", you can type "mkdir -p {a,b,c}" in the terminal.