Uday Chhina
A01210638

Assignment 1, Part 1

Part 1:

```
vagrant ~ → mkdir bin
```

```
vagrant ~/bin → pwd
/home/vagrant/bin
vagrant ~/bin →
```

```
# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi
```

Part 2:

Total 5 scripts, 2 scripts for project 1 and 1 for each of the remaining projects.

Script One, Project 1 (notes_cli_write)

```
#!/bin/bash
#: Title       : notes_cli_write
#: Date        : Oct 28 2022
#: Author      : Uday Chhina
#: Version     : 1.0
#: Description : Write notes.
#: Options     : None
```

Script Two, Project 1 (notes_cli_rs)

```
#!/bin/bash
#: Title       : notes_cli_rs
#: Date        : Oct 28 2022
#: Author      : Uday Chhina
#: Version     : 1.0
#: Description : Read and search for notes.
#: Options     : -t for searching with tags
#:               -m for searching with titles
#:               -d for searching with dates
```

Script Three, Project 2 (sys_info)

```
#!/bin/bash
#: Title       : sys_info
#: Date        : Oct 28 2022
#: Author      : Uday Chhina
#: Version     : 1.0
#: Description : Show system info.
#: Options     : -t for showing the tux ascii image art
```

Script Four, Project 3 (backup)

```
#!/bin/bash
#: Title       : backup
```

```
#: Date        : Oct 28 2022
#: Author      : Uday Chhina
#: Version     : 1.0
#: Description : Backup a directory or directories.
#: Options     : -d for giving the directories for backing up
#:               -b for the the directory to backup into
#:               -n to provide a name for the backup
```

Script Five, Project 4 (`mount_info`)

```
#!/bin/bash
#: Title       : mount_info
#: Date        : Oct 28 2022
#: Author      : Uday Chhina
#: Version     : 1.0
#: Description : Find if fs or devices are mounted to display information
#: Options     : None
```

Part 3:

Script One, Project 1 (`notes_cli_write`)

all notes will be kept in the same directory

Create directory for notes if it doesn't exist. The name of the directory is "/home/vagrant/all_notes"
Prompt the user for title of the note with the read utility
Store the title in a variable
Prompt the user to enter the tags separated with spaces. If this is empty, no tags will be added.
Prompt the user to write the note. Use the -d or -N options for read utility to get multiple lines of text. Store it in a variable. For the -N option, one extra prompt to get the max charactesr will be needed.
Get the current date with the date utility, formatted.
Assign the title and date to a variable which will be the name of the file that stores the text for this note.
Make the file with the new name and append the title formatted with two stars around it, (** Example **)
Append the space separated tags, if they exist, in a new line after the title (#tag1, #tag2, etc.) in the file.
Append the note text in the file on a new line. End with new line.

Script Two, Project 1 (`notes_cli_rs`)

default behavior for this script will be doing a "cat" on the provided note name (argument) and giving an error if the note wasn't found. This will essentially be the read functionality
when any of the 3 options are given, the script will search instead.
when option -t is given, the script will search the files in the directory with the grep utility, looking for the pattern for the title (** Title **). We will use the -l option with grep to only get the file names. This will be output for the user. The user can then use that to read the required note.
if the -m option is given, the we will use the find utility to find the names that match the given argument and output to the terminal. User can use that to read the required note.

if the -d option is given, the find utility will be used to search for files that match the given date string in the same format as what the write script saved it as. Again, this will be output to the user to that they may use the script to read the note that they want to.

Script Three, Project 2 (`sys_info`)

use command substitution for clearing the screen first. $(clear)
get the hostname with the hostname utility
use the whoami utility to get the user name.
read the /proc/uptime file to get the system uptime in seconds, format it into hours and minutes.
read the /etc/os-release file for the name of the OS.
use the uname utility to get the kernel version (uname -r)
use the $SHELL variable to get the name of the login shell.
use the apt list and the word count utilities/commands to get the total number of packages installed.
read the /proc/meminfo file to get the total memory and the memory used
hold these values in variables and print them to stdout when the script runs. Format the text using colors and whitespace.

when the script is ran with the -t option, print an ascii art of the tux which is stored in a file. We can use the cat utility to print it to the screen. Format it so that the text is before or after the tux.

Script Four, Project 3 (backup)

create the default directory if it doesn't exist.
for each of the directories given in the -d option, use the tar utility to back up all the directories in the default option.
if the -b option is given, create the dir if it doesn't exist and then back up the directories in that directory.
all of these will be given a default name with the current date and time. If the -n option is given, use that to name the backup files using the tar utility.

Script Five, Project 4 (`mount_info`)

use the findmnt utility to find if the device is mounted or not.
If it is mounted use the df utility (df -H) to get the size of the mounted system in human readable format.
if the device or filesystem is not mounted, give error message that it's not mounted.