1) Write a bash script to add 2 float numbers.

Code:

```
#!/bin/bash

# Function to add two float numbers
add_float() {
    local result=$(echo "$1 + $2" | bc -l)
    echo $result
}

# Main script
echo "Enter first float number: "
read num1

echo "Enter second float number: "
read num2

result=$(add_float $num1 $num2)
echo "The sum is: $result"
```

Output:

```
(base) go-d-code@gem-zangetsu:~/Roger/College$ bash "/home/go-d-code/Roger/College/OS_Lab_Codes/Lab – Exercise/1.sh"
Enter first float number:
9.2923
Enter second float number:
1.9283
The sum is: 11.2206
```

2) Write a bash script that monitors and displays the current CPU and memory usage of the system.

Code:

```
#!/bin/bash

# Function to display CPU and memory usage
monitor_usage() {
    echo "CPU Usage:"
    top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print 100 - $1"%"}'
    echo "Memory Usage:"
    free -m | awk 'NR==2{printf "Total: %s MB\nUsed: %s MB\nFree: %s MB\n", $2,$3,$4}'
}

# Main script
monitor_usage
```

Output:

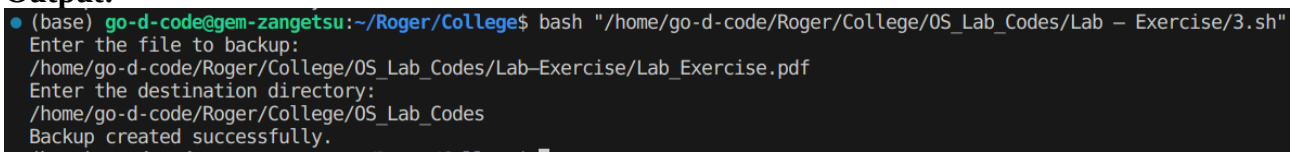
```
(base) go-d-code@gem-zangetsu:~/Roger/College$ bash "/home/go-d-code/Roger/College/OS_Lab_Codes/Lab – Exercise/2.sh"
CPU Usage:
28.4%
Memory Usage:
Total: 15685 MB
Used: 2199 MB
Free: 9427 MB
```

3) Write a bash script that takes a user-input file and destination directory, then creates a backup of the file in the specified destination.

Code:

```
#!/bin/bash

# Function to create a backup of a file
backup_file() {
    cp "$1" "$2"
    echo "Backup created successfully."
}

# Main script
echo "Enter the file to backup: "
read file

echo "Enter the destination directory: "
read destination

backup_file $file $destination
```

Output:



4) Write a user management script with options to add a new user, remove an existing user, and list all users on the system.

Code:

```
#!/bin/bash

# Function to add a new user
add_user() {
    echo "Enter username to add: "
    read username
    sudo adduser $username
}

# Function to remove an existing user
remove_user() {
    echo "Enter username to remove: "
    read username
    sudo deluser $username
}

# Function to list all users
list_users() {
    cut -d: -f1 /etc/passwd
}
```

```
# Main script
echo "User Management Script"
echo "1. Add a new user"
echo "2. Remove an existing user"
echo "3. List all users"
read choice

case $choice in
    1) add_user ;;
    2) remove_user ;;
    3) list_users ;;
    *) echo "Invalid option" ;;
esac
```

Output:

```
(base) go-d-code@gem-zangetsu:~/Roger/College$ bash "/home/go-d-code/Roger/College/OS_Lab_Codes/Lab-Exercise/4.sh"
User Management Script
1. Add a new user
2. Remove an existing user
3. List all users
1
Enter username to add:
testuser
[sudo] password for go-d-code:
Adding user `testuser' ...
Adding new group `testuser' (1001) ...
Adding new user `testuser' (1001) with group `testuser' ...
Creating home directory `/home/testuser' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for testuser
Enter the new value, or press ENTER for the default
        Full Name []: Test User
        Room Number []: 1
        Work Phone []: 1234567890
        Home Phone []: 1234567890
        Other []:
Is the information correct? [Y/n] y
```

```
(base) go-d-code@gem-zangetsu:~/Roger/College$ bash "/home/go-d-code/Roger/College/OS_Lab_Codes/Lab-Exercise/4.sh"
User Management Script
1. Add a new user
2. Remove an existing user
3. List all users
2
Enter username to remove:
testuser
Removing user `testuser' ...
Warning: group `testuser' has no more members.
Done.
```

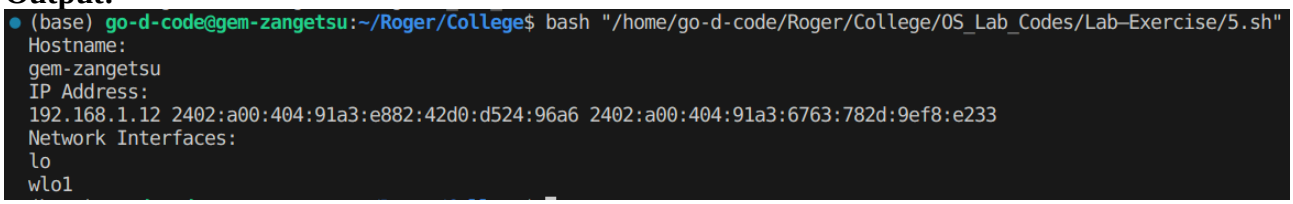
5) Write a script that retrieves and displays information about the network, including the hostname, IP address, and a list of network interfaces.

Code:

```
#!/bin/bash

# Function to retrieve network information
network_info() {
    echo "Hostname:"
    hostname
    echo "IP Address:"
    hostname -I
```

```
    echo "Network Interfaces:"
    ifconfig -a | grep -o '^[a-zA-Z0-9]*'
}

# Main script
network_info
```

Output:



6) Write a bash script that utilizes system calls to create a directory and a file within that directory.

Code:

```
#!/bin/bash

# Function to create directory and file
create_dir_and_file() {
    echo "Enter directory name: "
    read directory
    mkdir $directory
    echo "Enter file name: "
    read filename
    touch "$directory/$filename"
    echo "Directory '$directory' and file '$filename' created successfully."
}

# Main script
create_dir_and_file
```

Output: