LAB 8

Q1. Explain what the following commands do (with examples) and practice them:

- **lockfile** The lockfile command is used to create semaphore files. These files act as simple locks that prevent multiple processes from simultaneously accessing a shared resource or critical section.
  lockfile /tmp/my_lock_file.lock
- **cksum** Prints CRC checksum and byte counts of each FILE.
  cksum /home/Zuhaib /Desktop/CEW-OEL/main.c
- **comm** Compares two sorted files line by line.
  comm file1.txt file2.txt
- **csplit** Splits a file into sections determined by context lines
  csplit myfile.txt ‘a’
- **chattr** Changes file attributes on a Linux file system.
  chattr +i myfile.txt
- **touch** command in Linux is used to create empty files.
  touch example.txt

Q2. What do the following do:

- **cat ch1** Prints whatever is mentioned in the file ch1.
- **cat ch1 ch2 ch3 > “your-practical-group”** Concatenates the contents of three files (ch1, ch2, and ch3) into a new file named "your-practical-group" or simply “g3”
- **cat note5 >> notes** Appends the content of note5 to the end of the file named notes.
- **cat > temp1** This command is used to create or overwrite the content of a file named temp1.Whatever text we give to it after this command is overwritten in the file temp1
- **cat > temp2 << “Zuhaib”** This command creates or overwrites a file named temp2 and allows you to input multiple lines of text interactively. The content input continues until you type a line that matches the specified delimiter, in this case, "Zuhaib". The delimiter is used to signal the end of the input.

Q3. Practice the following commands and explain each:

- **cpio** The cpio command is used to copy files to and from archives. It's often used in combination with find to archive or copy files.
- **sort** The sort command is used to sort lines of text files. It can be used to sort files or the output of other commands.
- **fuser** The fuser command is used to identify processes using files or sockets. It can be helpful to find which processes are accessing a particular file or device.
- **file** The file command is used to determine the file type of a file. It examines the content and provides information about the file.

Q4. Differentiate between cp and cpio command?

The cp command is used for copying files and directories from one location to another. The cpio command is used for creating or extracting archives. It can copy files to or from an archive, making it suitable for backup purposes.

Q5. What does the z option of the tar command do? Explain with examples.

Tar is an archiving utility. It's -z option compresses or decompresses the archive through gzip.

For example:

```
tar -czvf archive.tar.gz directory_to_compress
```

- -c: Create a new archive
- -z: Compress the archive using gzip
- -v: Verbose mode (display progress)
- -f: Specify the name of the archive file

This command compresses the contents of directory_to_compress into a tar archive named archive.tar.gz.

Q6. Write two commands to take the backup of your home-folder and all sub-folders. The destination folder should be /home/bkup. NOTE: size of backup should be smaller than original folder.

Using gzip compression: tar -czvf /home/bkup/home_backup.tar.gz /home/Zuhaib

Using bzip2 compression: tar -cjvf /home/bkup/home_backup.tar.bz2 /home/Zuhaib

Q7. What is the difference between the permissions 777 and 775 of the chmod command?

The 777 permission allows complete read, write and execute rights to the owner, groups and others. Whereas the 775 permission allows complete read, write and execute rights to the owner and groups, but only read and execute command to others.

LAB 9

Q1: Write a shell program that takes one parameter (your name) and displays it on the screen.

```
#!/bin/bash

echo "$1"
```

Q2: Write a shell program that takes a number parameters equal to the last digit of your roll number and displays the values of the built in variables such as $#, $0, and $* on the screen.
#!/bin/bash

```
if [ "$#" -eq 3 ]; then

        echo "Number of arguments: $#"

        echo "Name of the script: $0"

        echo "All arguments as a single string: $*"

else

       echo "Please provide three arguments only"

fi
```

Q3: Write a script that displays the values of the following environment variables i) SHELL ii) PWD

```
#!/bin/bash

echo "PWD: $PWD"

echo "SHELL: $SHELL"
```

LAB 10

Q1: What test command should be used to test that /usr/bin is a directory or a file?

For file, use -f test command: [-f "/usr/bin"]

For directory, use -d test command: [-d "/usr/bin"]

Q2: Write a script that takes two strings as input compares them and depending upon the results of the comparison prints the results.

```
#!/bin/bash

echo "Enter the first string: "
read str1

echo "Enter the second string: "
read str2

if [ "$str1" = "$str2" ]; then
        echo "The strings are equal"

else
        echo "The strings are not equal"

fi
```

Q3: Write a script that takes a number (parameter) from 1-3 as input and uses case to display the name of corresponding month.

```
#!/bin/bash

case "$1" in

1)
        echo "January";;

2)
        echo "February";;

3)
        echo "March";;

*)
        echo "Only enter numbers from 1-3";;
esac
```

Q4: Write a script that calculates the average of all even numbers less than or equal to your roll number and prints the result.

```
#!/bin/bash

roll=1
sum=0
count=0

for ((i=2; i<=roll; i+=2)); do
        sum=$((sum + i))
        count=$((count + 1))
done

if [ "$count" -gt 0 ]; then

        average=$((sum/count))
        echo "The Average of even numbers upto $roll: $average"

else
        echo "There is no roll even number uptill your roll number :("

fi
```

Q5: Write a function that displays the name of the week days starting from Sunday if the user passes a day number. If a number provided is not between 1 and 7 an error message is displayed.

```
#!/bin/bash
func (){
        case "$1" in
        1)
                echo "Sunday";;
        2)
                echo "Monday";;
        3)
                echo "Tuesday";;
        4)
                echo "Wednesday";;

        5)
                echo "Thursday";;
        6)
                echo "Friday";;
        7)
                echo "Saturday";;

        *)
                echo "Only enter numbers from 1-7";;
        esac
}
func $1
```

Q6: Write scripts that displays the parameters passed along with the parameter number using while and until statements.

USING WHILE

```
        #!/bin/bash

        count=1

        while [ "$1" ]
        do
                echo "Argument # $count = $1 "
                count=$((count+1))
                shift
        done
```

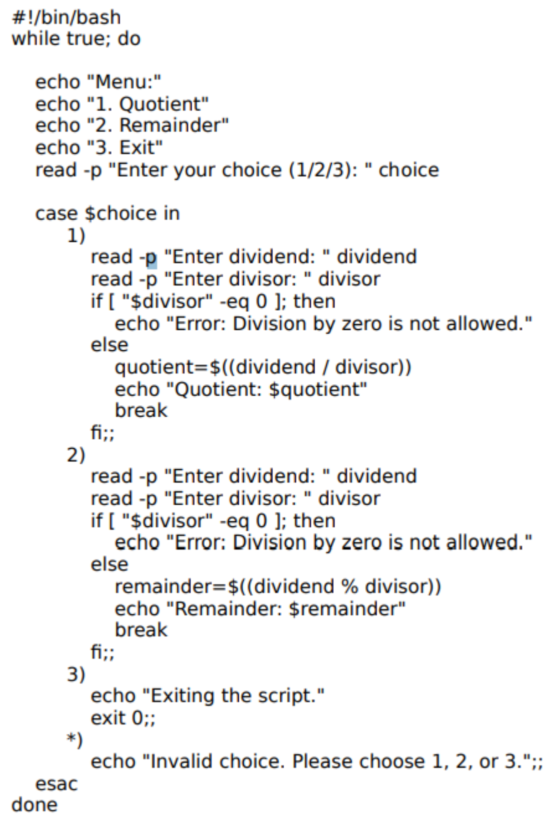
USING UNTIL

```
        #!/bin/bash

        count=1

        until [ ! "$1" ]
        do
                echo "Argument # $count = $1 "
                count=$((count+1))
                shift
        done
```

Q7: Write a script that displays the following menu:

- **Quotient**
- **Remainder**

Depending on user’s choice, the result of division must be displayed and the loop breaks. The two numbers (dividend and divisor) must be supplied at runtime as command line arguments. If user chooses an item that is not in the list, he must be prompted to make proper choice and the loop must restart (or continue).



```
#!/bin/bash
while true; do

   echo "Menu:"
   echo "1. Quotient"
   echo "2. Remainder"
   echo "3. Exit"
   read -p "Enter your choice (1/2/3): " choice

   case $choice in
       1)
          read -p "Enter dividend: " dividend
          read -p "Enter divisor: " divisor
          if [ "$divisor" -eq 0 ]; then
             echo "Error: Division by zero is not allowed."
          else
             quotient=$((dividend / divisor))
             echo "Quotient: $quotient"
             break
          fi;;
       2)
          read -p "Enter dividend: " dividend
          read -p "Enter divisor: " divisor
          if [ "$divisor" -eq 0 ]; then
             echo "Error: Division by zero is not allowed."
          else
             remainder=$((dividend % divisor))
             echo "Remainder: $remainder"
             break
          fi;;
       3)
          echo "Exiting the script."
          exit 0;;
       *)
          echo "Invalid choice. Please choose 1, 2, or 3.";;
   esac
done
```