LINUX PROGRAMMING

ASSIGNMENT-8

NAME: V S SAIDATTA USN: ENG24CY0175 ROLLNO:28

1. Write a shell script using if...else to check if a number is even or odd.

A.

```
#!/bin/bash

read -p "Enter a number: " num

# Check if the input is a valid integer
if ! [[ "$num" =~ ^-?[0-9]+$ ]]; then
        echo "Error: '$num' is not a valid integer."
        exit 1
fi

if (( num % 2 == 0 )); then
        echo "$num is even."
else
        echo "$num is odd."
Fi
```

2. Explain the difference between if and case statements in bash.

A.

'if statement' :

Used for evaluating expressions like:

- Numeric comparisons
- String comparisons
- File checks

- Command exit status

‘Case statement’:

Used for matching a single value against multiple patterns. It's simpler and cleaner than `if` when checking multiple fixed values or string patterns.

3. Write a script to find the largest of three numbers entered by the user.

A.

```
#!/bin/bash

read -p "Enter first number: " num1
read -p "Enter second number: " num2
read -p "Enter third number: " num3

if ! [[ "$num1" =~ ^-?[0-9]+$ && "$num2" =~ ^-?[0-9]+$ && "$num3" =~ ^-?[0-9]+$ ]]; then
        echo "Error: Please enter valid integers."
        exit 1
fi

if (( num1 >= num2 && num1 >= num3 )); then
        echo "The largest number is: $num1"
elif (( num2 >= num1 && num2 >= num3 )); then
        echo "The largest number is: $num2"
else
        echo "The largest number is: $num3"
fi
```

4. How do you use a for loop to traverse an array in bash? Give an example. The array is defined as arr=(123, “Abs”, -2.3, ‘A’, 23.56, 0).

A.

```
#!/bin/bash
```

```
# Define the array
arr=(123 "Abs" -2.3 'A' 23.56 0)

# Traverse using a for loop
echo "Traversing array elements:"
for element in "${arr[@]}"; do
        echo "$element"
done
```

5. Write a shell script to loop through all files in the current directory and display their names.

A.

```
#!/bin/bash

echo "Files in the current directory:"

for file in *; do
        if [ -f "$file" ]; then
                echo "$file"
        fi
done
```

6. What is the difference between while and until loops in bash?

A.

Feature	`while` Loop	`until` Loop
Condition True?	Loop runs while condition is true	Loop runs until condition is true
When to Use	When you want to run as long as a condition holds	When you want to run until a condition is met
Exit Condition	Exits when condition becomes false	Exits when condition becomes true

7. Write a countdown timer script using a while loop.

A.

```
#!/bin/bash

read -p "Enter countdown time in seconds: " time
if ! [[ "$time" =~ ^[0-9]+$ ]]; then
        echo "Invalid input. Please enter a positive integer."
        exit 1
fi
while [ $time -gt 0 ]; do
        echo -ne "Time remaining: $time seconds\r"
        sleep 1
 (      (time--))
done
echo -e "\n Time's up!"
```

8. How do you use break and continue statements in loops? Give examples.

A.

'Break': Used to exit a loop (for, while, or until) immediately.

CODE:

```
#!/bin/bash
for i in {1..10}
do
        if [ $i -eq 5 ]; then
                echo "Found 5, breaking the loop."
        break
        fi
        echo "Number: $i"
Done
```

'continuous': Skips the current iteration and proceeds to the next one.

CODE:

```
#!/bin/bash
```

```
for i in {1..5}
do
        if [ $i -eq 3 ]; then
                echo "Skipping 3"
                continue
        fi
        echo "Number: $i"
Done
```

9. Write a script to check if a file exists or not using the if and else loop.

A.

CODE:

```
#!/bin/bash

filename="testfile.txt"
if [ -e "$filename" ]; then
        echo "File '$filename' exists."
else
        echo "File '$filename' does not exist."
Fi
```

10. Write a script to calculate factorial of a number using for loop.

A.

CODE:

```
#!/bin/bash

read -p "Enter a positive integer: " num
factorial=1

if ! [[ "$num" =~ ^[0-9]+$ ]]; then
        echo "Error: Please enter a valid non-negative integer."
```

```
        exit 1
fi

for (( i=1; i<=num; i++ ))
        do
                factorial=$((factorial * i))
        done
echo "Factorial of $num is $factorial"
```