School Of Engineering

Linux Programming
Assignment-9

Name: Neehara Lakshmi
USN No: ENG24CY0138
Roll No: 55
Section: B (CyberSecurity)
Semester: 3rd

Q1. Write a shell script using if...else to check if a number is even or odd.

Ans:

Example Code:

```
#!/bin/bash
echo "Enter a number:"
read num
if [ $((num % 2)) -eq 0 ]; then
    echo "$num is even"
else
    echo "$num is odd"
fi
```

Output:

```
Enter a number:
7
7 is odd
```

Q2. Explain the difference between if and case statements in bash.

Ans:

‘ if ’ Statement:

Condition-based - Uses boolean expressions and comparisons

Flexible conditions - Can use complex logical operators (`&&`, `||`)

Range checking - Good for numerical ranges and string patterns

Multiple conditions - Handles `elif` for multiple test cases

Example Code:

```
if [ $age -ge 18 ]; then
    echo "Adult"
elif [ $age -ge 13 ]; then
    echo "Teen"
else
    echo "Child"
fi
```

‘ case ’ Statement:

Pattern-based- Matches against specific patterns/values

Exact matching- Best for discrete values rather than ranges

Cleaner syntax- More readable for multiple exact matches

Wildcard support- Can use `*`, `?` in patterns

Example Code:

```
case $option in
    "start")
        echo "Starting service"
```

```
        ;;
    "stop")
        echo "Stopping service"
        ;;
    "restart")
        echo "Restarting service"
        ;;
    *)
        echo "Invalid option"
        ;;
esac
```

Q3. Write a script to find the largest of three numbers entered by the user.

Ans:

Example Code:

```
#!/bin/bash
echo "Enter three numbers:"
read num1 num2 num3
if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]; then
    echo "Largest number is: $num1"
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]; then
    echo "Largest number is: $num2"
else
    echo "Largest number is: $num3"
fi
```

Output:

```
Enter three numbers:
12 25 8
Largest number is: 25
```

Q4. 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).

Ans: To traverse an array in bash using a for loop, you can use the ‘ "${array[@]}" ’ syntax to access all elements. Here's an example script based on the array you provided:

Example Code:

```
#!/bin/bash
# Define the array without commas (bash arrays are space-separated)
arr=(123 "Abs" -2.3 'A' 23.56 0)
# Loop through each element in the array
for element in "${arr[@]}"; do
    echo "$element"
done
```

Output:

123
Abs
-2.3
A
23.56
0

Q5. Write a shell script to loop through all files in the current directory and display their names

Ans:

Example Code:

```
#!/bin/bash
echo "Files in current directory:"
# Loop through all files and directories
for file in *; do
    echo "$file"
done
```

Output:

Files in current directory:
document.txt
script.sh
images/
backup.tar.gz
config.json

Q6. What is the difference between while and until loops in bash?

Ans:

‘ while ’ Loop:

Runs while condition is True.
Continues execution as long as the condition evaluates to true.
Stops when condition becomes false.

Syntax:

```
while [ condition ]; do
    commands
done
```

Example Code:

```
count=1
while [ $count -le 5 ]; do
    echo "Count: $count"
    ((count++))
```

```
done
```

‘ until ’ Loop:

Runs until condition becomes True.

Continues execution as long as the condition evaluates to false.

Stops when condition becomes true.

Syntax:

```
until [ condition ]; do
    commands
done
```

Example Code:

```
count=1
until [ $count -gt 5 ]; do
    echo "Count: $count"
    ((count++))
done
```

Q7. Write a countdown timer script using a while loop.

Ans:

Example Code:

```
#!/bin/bash
echo "Enter countdown time in seconds:"
read seconds
while [ $seconds -gt 0 ]; do
    echo -ne "Time remaining: $seconds seconds\033[0K\r"
    sleep 1
    ((seconds--))
done
echo -e "\nTime's up!"
```

Q8. How do you use break and continue statements in loops? Give examples.

Ans:

‘break’ Statement:

Exits the loop immediately

Used to terminate the loop before its normal completion

Example Code:

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

```
done
echo "Loop exited"
```

Output:

```
Number: 1
Number: 2
Number: 3
Number: 4
Loop exited
```

'continue' Statement:

Skips the current iteration and continues with the next one

Used to bypass specific iterations without exiting the loop

Example Code:

```
#!/bin/bash
for i in {1..10}; do
    if [ $((i % 2)) -eq 0 ]; then
        continue
    fi
    echo "Odd number: $i"
done
```

Output:

```
Odd number: 1
Odd number: 3
Odd number: 5
Odd number: 7
Odd number: 9
```

Q9. Write a script to check if a file exists or not using the if and else loop.

Ans :

Example Code:

```
#!/bin/bash
echo "Enter the filename:"
read filename
if [ -f "$filename" ]; then
    echo "File '$filename' exists."
else
    echo "File '$filename' does not exist."
fi
```

Output:

```
Enter the filename:
script.sh
File 'script.sh' exists.
```

Q10. Write a script to calculate factorial of a number using for loop. (

Ans:

Example Code:

```
#!/bin/bash
echo "Enter a number:"
read num
factorial=1
if [ $num -lt 0 ]; then
    echo "Factorial is not defined for negative numbers."
else
    for (( i=1; i<=num; i++ )); do
        factorial=$((factorial * i))
    done
    echo "Factorial of $num is: $factorial"
fi
```

Output:

```
Enter a number:
5
Factorial of 5 is: 120
```

THANK YOU