No worries, Prasanth! I've got you covered from the **very basics**. Let's build your understanding of **UNIX shell scripting** step by step so you can write and understand all those scripts confidently.

🧠 What is a Shell Script?

A **shell script** is a file containing a sequence of **commands** that the shell (like Bash) can run.

✅ Basic Setup:

1. **File Name**: Usually ends with .sh, like myscript.sh
2. **Shebang Line**: Always start your script with:
3. #!/bin/bash

This tells the system to use **bash shell** to run the script.

📘 Basic Syntax of Shell Script

1. Declaring Variables

name="Prasanth"

age=21

(No spaces before or after =)

2. Using Variables

echo "My name is $name and I am $age years old"

3. Taking Input

echo "Enter your name:"

read username

echo "Hello, $username"

4. If-Else

if [$age -gt 18]

then

  echo "Adult"

else

  echo "Minor"

fi

5. Loops

For Loop:

```
for i in 1 2 3
do
  echo "Number $i"
done
```

While Loop:

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

🛠 How to Write and Run a Shell Script

Step 1: Create a file

nano script.sh

Paste your code there.

Step 2: Make it executable

chmod +x script.sh

Step 3: Run it

./script.sh

✅ Sample Script: Add Two Numbers

```
#!/bin/bash

echo "Enter first number:"
read a

echo "Enter second number:"
read b
```

```
sum=$((a + b))

diff=$((a - b))

prod=$((a * b))

quot=$((a / b))


echo "Sum = $sum"

echo "Difference = $diff"

echo "Product = $prod"

echo "Quotient = $quot"
```

Let me know which of your 15 questions you'd like to write first, and I'll walk you through that script **step-by-step like a pro teacher**!

Here are the solutions and separate explanations for each shell scripting question for your UNIX lab internal exam:

---

1. Shell Script to Accept Two Numbers and Perform All Arithmetic Operations

```
#!/bin/bash

echo "Enter two numbers:"

read a b

echo "Addition: $((a + b))"

echo "Subtraction: $((a - b))"

echo "Multiplication: $((a * b))"

echo "Division: $((a / b))"

echo "Modulus: $((a % b))"
```

Explanation:

- read a b: Accepts two numbers from the user.
- $((expression)): Used for arithmetic calculations.

---

2. Largest of Three Numbers Using Conditional Execution Operators

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

```
echo "Enter three numbers:"
read a b c
[ $a -ge $b ] && [ $a -ge $c ] && echo "$a is largest" || \
([ $b -ge $a ] && [ $b -ge $c ] && echo "$b is largest" || echo "$c is largest")
```

Explanation:

- && executes the next command only if the previous is true.
- || executes if the previous fails.

---

3. Test File Permissions

```
#!/bin/bash
echo "Enter filename:"
read file

[ -x "$file" ] && echo "$file is executable"
[ -r "$file" ] && echo "$file is readable"
[ -w "$file" ] && echo "$file is writable"
[ -r "$file" ] && [ -w "$file" ] && echo "$file is readable and writable"
```

Explanation:

- -x, -r, -w: Check if file is executable, readable, or writable.

---

4. Show Last Login User and Terminal

```
#!/bin/bash
last -1 | awk '{print "User: "$1", Terminal: "$2}'
```

Explanation:

- last -1: Shows the most recent login.
- awk: Used to extract user and terminal name.

---

5. Count Number of Files in a Directory

```
#!/bin/bash
echo "Enter directory path:"
read dir
```

```
count=$(ls -l "$dir" | grep ^- | wc -l)
echo "Number of files: $count"
```

Explanation:

- grep ^-: Filters regular files.
- wc -l: Counts lines.

---

6. Pattern Print

```
#!/bin/bash
for ((i=1;i<=4;i++))
do
 for ((j=1;j<=i;j++))
 do
  echo -n "$j"
 done
 echo
done
```

Explanation:

- Nested loops print increasing sequences.

---

7. Days in Given Month and Year

```
#!/bin/bash
echo "Enter month (MM):"
read m
echo "Enter year (YYYY):"
read y
cal $m $y | grep -v '[a-zA-Z]' | wc -w
```

Explanation:

- cal: Prints calendar.
- wc -w: Counts days.

---

8. Perfect Number Checker

```
#!/bin/bash
echo "Enter a number:"
read n
sum=0
for ((i=1;i<n;i++))
do
 if ((n%i==0))
 then
  sum=$((sum+i))
 fi
done
if ((sum==n))
then
 echo "$n is a perfect number"
else
 echo "$n is not a perfect number"
fi
```

Explanation:

- A perfect number equals the sum of its proper divisors.

---

9. String Concatenation via Arguments

```
#!/bin/bash
s1=$1
s2=$2
echo "Concatenated string: $s1$s2"
```

Explanation:

- $1 and $2: First and second command-line arguments.

---

10. Break and Continue

```
#!/bin/bash
for ((i=1;i<=10;i++))
```

```
do
 if ((i==5)); then
  continue
 fi
 if ((i==8)); then
  break
 fi
 echo $i
done
```

Explanation:

- continue: Skips current iteration.
- break: Exits loop.

---

11. Menu with Options

```
#!/bin/bash
while true
do
 echo "1. Current Directory"
 echo "2. Today's Date"
 echo "3. Logged in Users"
 echo "4. Quit"
 read -p "Enter choice: " ch
 case $ch in
  1) pwd ;;
  2) date ;;
  3) who ;;
  4) exit ;;
  *) echo "Invalid option" ;;
 esac
done
```

Explanation:

- case handles menu choices.
- while true: Keeps menu running until exit.

---

12. Delete Zero Byte Files

```
#!/bin/bash
find . -type f -size 0 -exec rm {} \;
```

Explanation:

- find: Searches for files.
- -size 0: Matches empty files.
- -exec rm: Deletes them.

---

13. Reverse Argument List

```
#!/bin/bash
for ((i=$#;i>0;i--))
do
 eval echo \${$i}
done
```

Explanation:

- $#: Number of arguments.
- eval: Used to reference variable names dynamically.

---

14. Factorial from Arguments

```
#!/bin/bash
for num in "$@"
do
 fact=1
 for ((i=1;i<=num;i++))
 do
  fact=$((fact*i))
 done
 echo "Factorial of $num is $fact"
```

done

Explanation:

- $@: All command-line arguments.
- Inner loop calculates factorial.

---

15. Time-Based Greeting

```
#!/bin/bash

hour=$(date +%H)


if [ $hour -ge 5 ] && [ $hour -lt 12 ]

then

  echo "Good Morning"

elif [ $hour -ge 12 ] && [ $hour -lt 17 ]

then

  echo "Good Afternoon"

elif [ $hour -ge 17 ] && [ $hour -lt 21 ]

then

  echo "Good Evening"

else

  echo "Good Night"

fi
```

Explanation:

- date +%H: Gets current hour.
- Conditions decide the greeting.

Let me know if you want a PDF or formatted notes for practice.

Absolutely! Here's a **detailed explanation of the shell scripting topics** involved in each question of your UNIX lab internal. These will help you understand not just the "how" but also the "why".

✅ **1. Accept Two Numbers & Perform Arithmetic Operations**

Topics Involved:

- **Variables**: Used to store user inputs.
- **Read Command**: Accepts user input.
- **Arithmetic Expansion**: $(()) is used to perform arithmetic operations.

Key Operators:

- +: Addition
- -: Subtraction
- *: Multiplication
- /: Division (integer only)
- %: Modulus (remainder)

✅ 2. Largest of Three Numbers Using Conditional Operators

Topics Involved:

- **Conditional Execution**:
  - &&: Executes next command only if the previous succeeds.
  - ||: Executes next command only if the previous fails.
- **Test Conditions**: [$a -ge $b] checks if a is greater or equal to b.

✅ 3. File Property Tests

Topics Involved:

- **File Test Operators**:
  - -r: Readable
  - -w: Writable
  - -x: Executable
- These tests are essential for checking access permissions in scripts.

✅ 4. Display Last Login User and Terminal

Topics Involved:

- **last command**: Displays login history.
- **awk command**: Text processing to extract specific fields (user, terminal).

✅ 5. Count Files in Directory

Topics Involved:

- **ls -l**: Lists files with details.
- **grep ^-**: Filters only regular files.
- **wc -l**: Counts number of lines = number of files.

✅ 6. Pattern Printing (Looping Concepts)

Topics Involved:

- **Nested Loops**:
    - for loop: Used to generate increasing sequences.
    - echo -n: Avoids newline to print in same line.
- Useful for understanding control structures.

✅ 7. Days in a Month and Year

Topics Involved:

- **cal command**: Displays calendar.
- **grep -v '[a-zA-Z]'**: Removes header.
- **wc -w**: Counts number of words = number of days.

✅ 8. Perfect Number Check

Topics Involved:

- **Loops and Conditions**:
    - Finds factors and adds them.
- **Perfect number**: A number that equals the sum of its divisors (excluding itself).

✅ 9. String Concatenation Using Arguments

Topics Involved:

- **Command Line Arguments**:
    - $1, $2: Refer to first and second arguments.
- Concatenation simply means printing them together.

✅ 10. break and continue Statements

Topics Involved:

- **break**: Exits the loop entirely.
- **continue**: Skips the current iteration, continues to next.
- Used for flow control in loops.

✅ 11. Menu-Driven Script

Topics Involved:

- **case statement**: Executes different blocks based on user input.
- **while true**: Keeps showing the menu until the user chooses to quit.
- Useful in automation scripts and interactive tools.

✅ 12. Delete Empty Files

Topics Involved:

- **find command**:
    - -type f: Files only
    - -size 0: Files of zero size
    - -exec: Executes a command (like rm) on each result.

✅ 13. Reverse Argument List

Topics Involved:

- **Shell Parameters**:
    - $#: Total number of arguments.
    - eval echo \${$i}: Dynamically accesses argument variables.

✅ 14. Factorial from Arguments

Topics Involved:

- **Positional Parameters**: $@ is a list of all arguments.
- **Loops**: Used to multiply numbers up to n for factorial.

✅ 15. Time-Based Greeting

Topics Involved:

- **Date Command**: date +%H gets current hour.
- **If-Else Statements**: Decide output based on time.

🌟 **Summary of Key Shell Scripting Concepts:**

Topic	Description
read	Accepts user input
echo	Displays output
$(())	Arithmetic operations
if, else, elif	Decision making
&&, `	
for, while loops	Repeated execution
case	Menu-like decision making
File Test Operators	Check file properties (-r, -w, -x)
Positional Parameters	$1, $2, $@, $#
break, continue	Flow control in loops
External Commands	cal, find, last, date, who, etc.

Let me know if you want this as a **PDF**, **practice Q&A**, or if you'd like **interactive quiz questions** to test yourself!