Shell Scripting Lab Manual

Part 1: Echo

Concept:

`echo` is used to display text or output to the terminal.

Syntax:

```
echo [string]
```

Example:

```
#!/bin/bash
echo "Hello, Shell Scripting!"
```

Exercise:

1. Write a script that prints your name using `echo`.
2. Modify the script to print your name on the first line and a greeting message on the second line.

Part 2: Read

Concept:

The `read` command is used to take input from the user during script execution.

Syntax:

```
read variable_name
```

Example:

```
#!/bin/bash
echo "Enter your name:"
read name
echo "Hello, $name!"
```

Exercise:

1. Write a script that asks for the user's age and then prints a message like: "You are [age] years old."
2. Modify the script to accept a favorite color from the user and print a personalized message like: "Your favorite color is [color]."

Part 3: Variable Declaration

Concept:

Variables in shell scripting are used to store data. By default, shell variables are untyped and can store any data type.

Syntax:

```
variable_name=value
```

Example:

```
#!/bin/bash
name="Alice"
age=25
echo "Name: $name"
echo "Age: $age"
```

Exercise:

1. Write a script to declare two variables, `num1` and `num2`, and assign them integer values. Then print their values.
2. Modify the script to print the sum of `num1` and `num2`.

Part 4: Summation

Concept:

To perform arithmetic operations such as addition, subtraction, multiplication, and division, you need to use `$(( ))` syntax in shell scripting.

Syntax:

```
result=$((num1 + num2))
```

Example:

```
#!/bin/bash
echo "Enter two numbers:"
read num1
read num2
sum=$((num1 + num2))
echo "The sum of $num1 and $num2 is: $sum"
```

Exercise:

1. Write a script that accepts two numbers from the user and calculates their sum.
2. Modify the script to calculate the product (multiplication) of the two numbers.
3. Extend the script to handle subtraction and division as well.

Part 5: Checking Equality Using If-Else Logic

Concept:

`if-else` statements are used for decision-making. We can check for equality or other conditions using operators.

Syntax:

```
if [ condition ]; then
   # Commands to execute if condition is true
else
   # Commands to execute if condition is false
fi
```

- `-eq`: checks if two values are equal.
- `-ne`: checks if two values are not equal.
- `-lt`: checks if the first value is less than the second.
- `-gt`: checks if the first value is greater than the second.

Example:

```
#!/bin/bash
echo "Enter a number:"
read num
if [ $num -eq 10 ]; then
   echo "You entered 10!"
```

```
else
   echo "You did not enter 10."
fi
```

Exercise:

1. Write a script that asks for a number and checks if it is equal to 100. Print a message accordingly.
2. Modify the script to check if the number is greater than or less than 100, and print different messages for each case.

Final Exercise: Combined Concepts

Scenario:

Write a shell script that does the following:

1. Prompts the user for their first name, last name, and age.
2. Displays a greeting message using `echo`.
3. Calculates and displays the sum of two numbers entered by the user.
4. If the sum is greater than 100, display "The sum is large." Otherwise, display "The sum is small."
5. Check if the age is 18 or greater. If true, print "You are an adult."; otherwise, print "You are a minor."

Submission Guidelines:

- Save your scripts with a `.sh` extension (e.g., `script.sh`).
- Submit your scripts to the instructor for grading.

Additional Resources:

1. Bash Scripting Guide
2. Shell Scripting Tutorial
