Bash Practice Questions for Chapter 9: Using Functions in Bash

The ninth chapter of the Bash Beginner Series on Linux Handbook focuses about using if-else, nested if else and case statements in bash scripts. These practice questions will help you practice what you learned in this chapter.

Exercise 1: Is it +ve?

Let's start with something simple. Write a bash shell script that checks if a given number is +ve or not. Use a function in your script.

Difficulty level: Easy

Hint: Use read command to accept input from user

Exercise 2: Functional multiplication table

Write a bash shell script that prints the multiplication table of a given number. Using functions is a must.

Difficulty level: Easy

Hint: Use for loop for a more effective script.

Exercise 3: Number is in its prime?

Write a shell script that accepts a number and check if it is prime or not. Using function is must here.

Difficulty level: Intermediate

Hint: Prime numbers are only divisible by 1 and the number itself. 23 is prime, 21 is not prime. 1 and 2 are also considered prime.

Exercise 4: Fibonacci series

Write a shell script that accepts a number greater than 3 and prints the Fibonacci sequence up to that term. DO NOT USE RECURSIVE FUNCTION.

Difficulty level: Intermediate to hard

Hint: Fibonacci series starts with 0 and 1 and the next term is the sum of the previous two terms.

Output: Check the result with 10 terms which should print 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Exercise 5: Fibonacci using recursion

Write a shell script that accepts a number greater than 3 and prints the Fibonacci sequence up to that term. But you MUST USE RECURSIVE FUNCTION here.

Difficulty level: Hard

Hint: Fibonacci series starts with 0 and 1 and the next term is the sum of the previous two terms.

Output: Check the result with 10 terms which should print 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Solutions to the Exercises

Solution 1: Functional multiplication table

> Write a bash shell script that checks if a given number is +ve or not. Use a function in your script.

```
#!/bin/bash

read -p "Enter number: " num

ispositive () {
    if [ "$1" -le 0 ]; then
        echo "Number $1 is negative"
    else
        echo "Number $1 is positive"
    fi
}
ispositive $num
```

Solution 2: Functional multiplication table

> Write a bash shell script that prints the multiplication table of a given number. Using functions is a must.

```
#!/bin/bash

read -p "Enter number: " num

table () {
    for i in {1..10}; do
        echo $(($1*$i))
    done
}
table $num
```

Solution 3: Number is in its prime?

> Write a shell script that accepts a number and check if it is prime or not. Using function is must here.

```
#!/bin/bash

read -p "Enter number: " num

isprime () {
    for((i=2; i<=num/2; i++)); do
        if [ $((num%i)) -eq 0 ]; then
            echo "$num is not a prime number."
            exit
        fi
    done
    echo "$num is prime number."
}

isprime $num
```

Solution 4: Fibonacci series

> Write a shell script that accepts a number greater than 3 and prints the Fibonacci sequence up to that term.

```
#!/bin/bash

read -p "Enter number greater than 3: " num

if [ "$num" -lt 3 ]; then
    echo "Number must be greater than 3"
    exit
fi

#initialize first and second terms
t1=0
t2=1

#initialize the next term (3rd term)
nextTerm=$(($t1 + $t2))

echo -n "$t1, $t2"

fibo () {
    for ((i = 3; i <= $num; ++i)); do
        echo -n ", $nextTerm";
        t1=$t2;
        t2=$nextTerm;
        nextTerm=$(($t1 + $t2));
    done
    echo ""
}

fibo $num
```

Solution 5: Fibonacci using recursion

> Write a shell script that accepts a number greater than 3 and prints the Fibonacci sequence up to that term. But you MUST USE RECURSIVE FUNCTION here.

```
#!/bin/bash

read -p "Enter number greater than 3: " num

if [ "$num" -lt 3 ]; then
    echo "Number must be greater than 3"
    exit
fi

echo -n "Fibonacci series for $num terms is: "

fibo() {
    if [ "$1" -le 1 ]; then
        echo -n "$1"
    else
        echo -n $(( $(fibo $(($1 - 1)) ) + $(fibo $(($1 - 2)) ) ))
    fi
}

for (( i=0; i<$num; ++i)); do
    fibo $i
    # Print , and space between the numbers except the last one
    if [ "$i" -lt $(($num -1)) ]; then
        echo -n ", "
    fi
done

echo ""
```