# LAB3 – Modified Script

---

## 📝 What This Assignment Was About

We had to take an existing script (`print_numbers.sh`) that just printed numbers from 1 to 5 and improve it so that it takes input from the user — specifically, the **start**, **end**, and **step** values. The script also had to check if the step is a positive number.

---

## 🔧 What The Original Script Did

The original script was very simple. It just printed numbers from 1 to 5:

```
#!/bin/bash

for i in {1..5}
do
  echo "Number: $i"
done
It didn't take any input or check anything. You couldn't change the range without
editing the file.

✨ What I Changed – New Script
I made a new script called enhanced_numbers.sh. This one lets the user type in 3
values:

Where to start

Where to end

The step size

It also checks if the step is greater than 0 before running.

💻 New Script: enhanced_numbers.sh
bash

#!/bin/bash

# Check for 3 arguments
if [ $# -ne 3 ]; then
  echo "Usage: $0 start end step"
  exit 1
fi

start=$1
end=$2
step=$3
```

```
# Make sure step is positive
if [ $step -le 0 ]; then
  echo "Error: Step must be a positive number."
  exit 1
fi

# Print the numbers
for (( i=$start; i<=$end; i+=$step ))
do
  echo "Number: $i"
done
```

🩹 Example Outputs

☑ Example 1

bash

```
$ ./enhanced_numbers.sh 1 10 2
Output:

Number: 1
Number: 3
Number: 5
Number: 7
Number: 9
```

☑ Example 2

bash

```
$ ./enhanced_numbers.sh 5 20 5
Output:

Number: 5
Number: 10
Number: 15
Number: 20
```

❌ Invalid Step

bash

```
$ ./enhanced_numbers.sh 1 10 -2
Output:

typescript

Error: Step must be a positive number.
```

❓ Extra Questions

Q1: What’s the difference between $1, $@, and $#?

$1 is the first argument

$@ is all the arguments

$# is the number of arguments

Example:

bash

```
$ ./script.sh apple banana cherry
$1 = apple
$@ = apple banana cherry
$# = 3
Q2: What does exit 1 mean?
exit 1 ends the script and tells the system something went wrong.
It’s used to stop the script when there’s an error like missing input or bad data.

exit 0 = everything was fine

exit 1 (or any non-zero number) = there was an error

☑ What I Learned
How to use $1, $@, and $#

How to check input in Bash scripts

How to use loops with custom values

That exit 1 helps stop the script when needed




---
```