Experiment 6: Shell Loops

Name:Tanishq bansal Roll No.: 590029349 Date: 2025-10-28

Aim:

- To understand and implement shell loops (`for`, `while`, `until`) in Bash.
- To practice loop control constructs (`break`, `continue`) and loop-based file processing.

Requirements

- A Linux system with bash shell.
- A text editor (nano, vim) and permission to create and execute shell scripts.

Theory

Loops allow repeated execution of commands until a condition is met. Common loop constructs in Bash include `for` (iterate over items), `while` (repeat while condition true), and `until` (repeat until condition becomes true). Loop control statements like `break` and `continue` change the flow inside loops. Loops are essential for automating repetitive tasks such as processing multiple files, generating sequences, and collecting user input.

Procedure & Observations

Exercise 1: Simple `for` loop

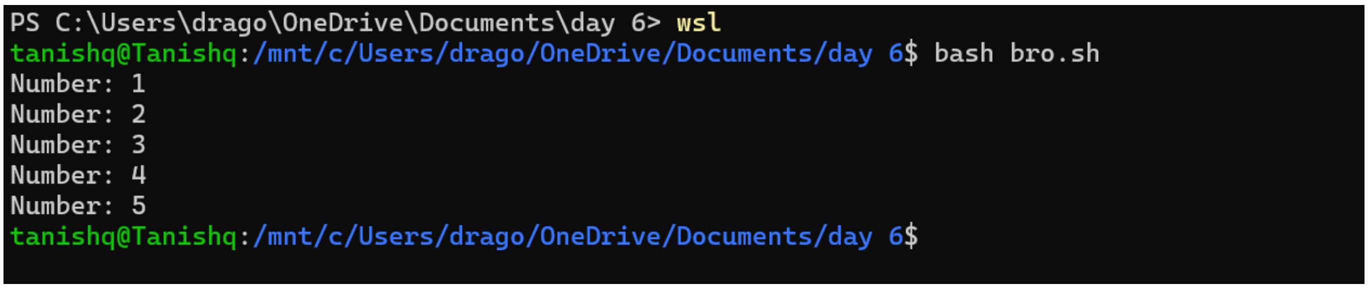
Task Statement:

Write a `for` loop that prints numbers 1 to 5.

Command(s):

```bash
for i in 1 2 3 4 5; do
  echo "Number: $i"
done
```

Output:



```
PS C:\Users\drago\OneDrive\Documents\day 6> wsl
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ bash bro.sh
Number: 1
Number: 2
Number: 3
Number: 4
Number: 5
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$
```

Exercise 2: `for` loop over files

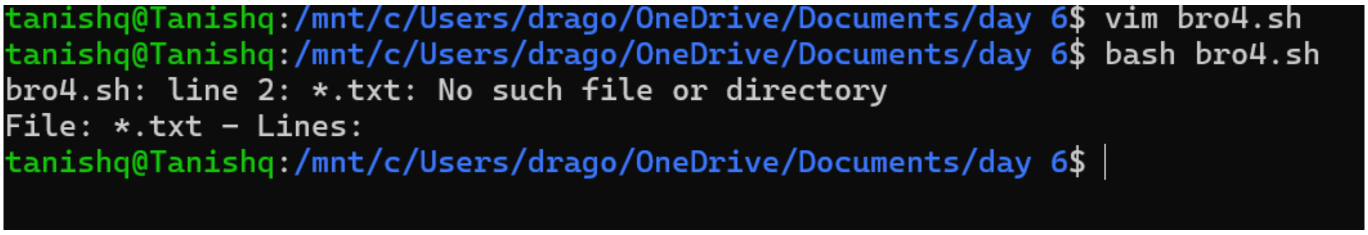
Task Statement:

Process all `.txt` files in a directory and count lines in each.

Command(s):

```
for f in *.txt; do
  echo "File: $f - Lines: $(wc -l < "$f")"
done
```

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ vim bro4.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ bash bro4.sh
bro4.sh: line 2: *.txt: No such file or directory
File: *.txt - Lines:
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$
```

Exercise 3: C-style `for` loop

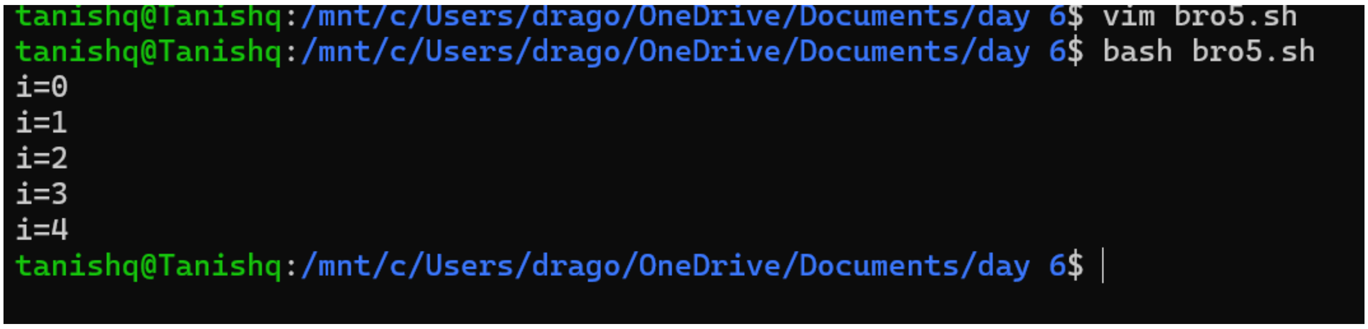
Task Statement:

Use arithmetic C-style loop for numeric iteration.

Command(s):

```
for ((i=0;i<5;i++)); do
  echo "i=$i"
done
```

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ vim bro5.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ bash bro5.sh
i=0
i=1
i=2
i=3
i=4
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$
```

Exercise 4: `while` loop and reading input

Task Statement:

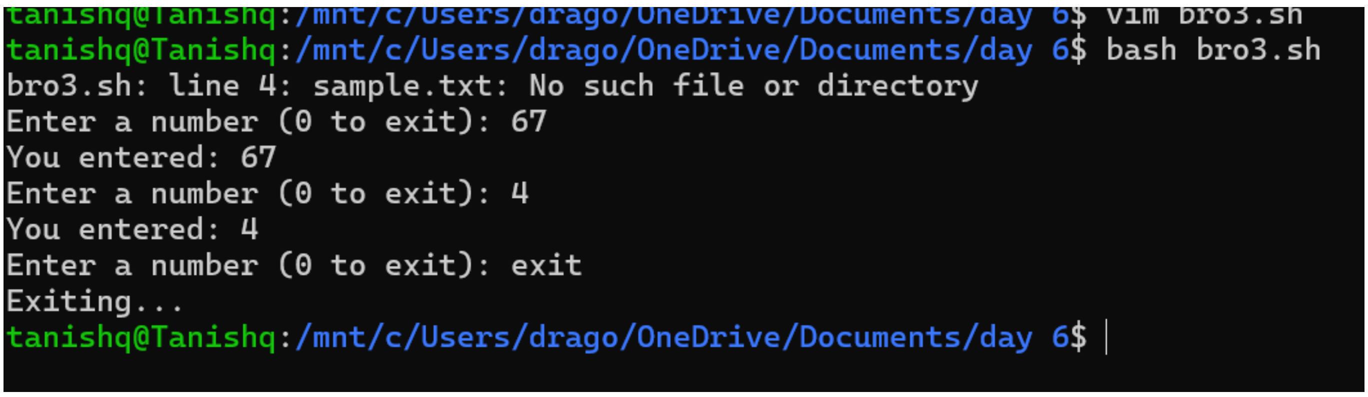
Write a `while` loop that reads lines from a file or from user input.

Command(s):

```
# Read from file
while read -r line; do
  echo "Line: $line"
done < sample.txt

# Read from user with exit condition
while true; do
  read -p "Enter a number (0 to exit): " n
  if [[ $n -eq 0 ]]; then
    echo "Exiting..."; break
  fi
  echo "You entered: $n"
done
```

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ vim bro3.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ bash bro3.sh
bro3.sh: line 4: sample.txt: No such file or directory
Enter a number (0 to exit): 67
You entered: 67
Enter a number (0 to exit): 4
You entered: 4
Enter a number (0 to exit): exit
Exiting...
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$
```

Exercise 5: `until` loop

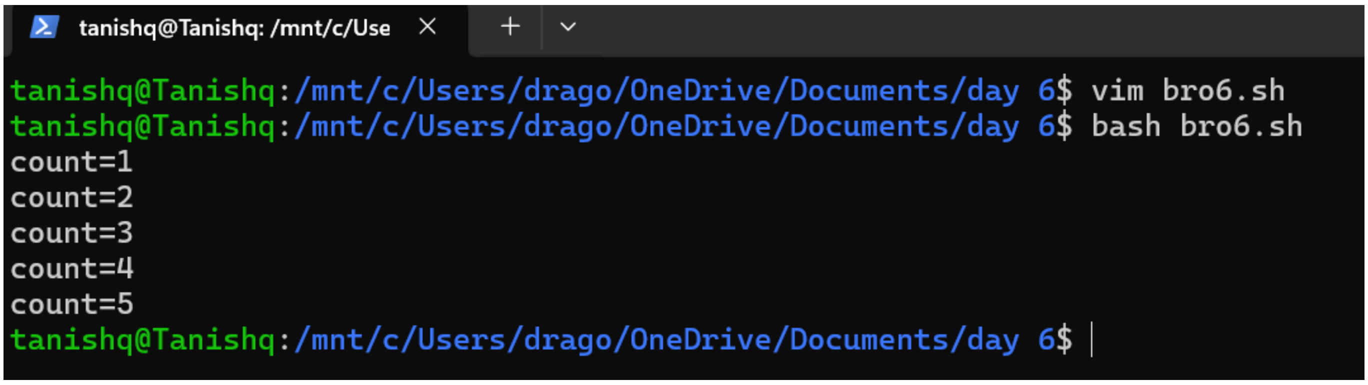
Task Statement:

Use an `until` loop to run until a condition becomes true.

Command(s):

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

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ vim bro6.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ bash bro6.sh
count=1
count=2
count=3
count=4
count=5
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$
```

Exercise 6: break and continue

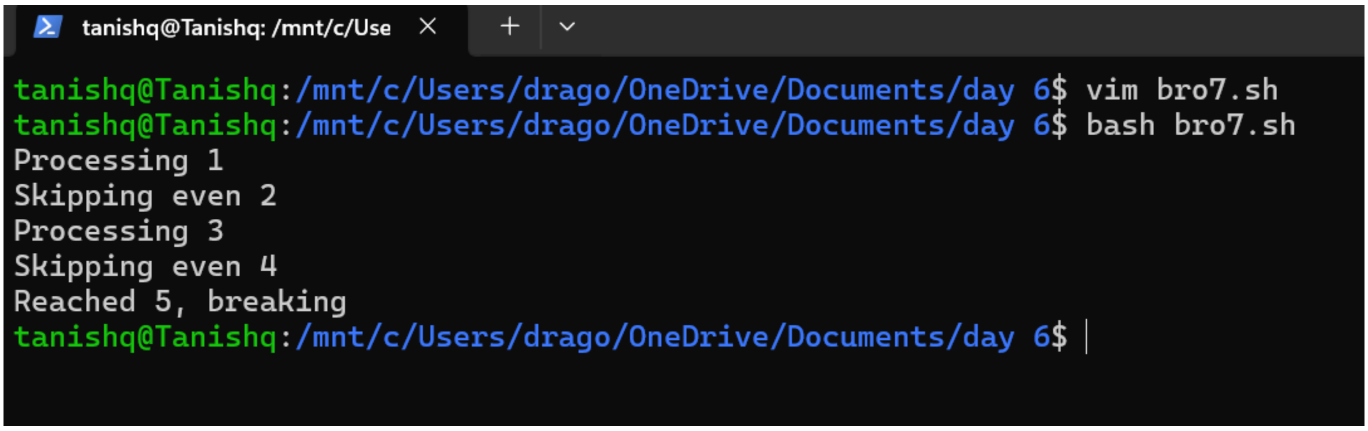
Task Statement:

Demonstrate break and continue inside a loop.

Command(s):

```
for i in {1..10}; do
  if [[ $i -eq 5 ]]; then
    echo "Reached 5, breaking"; break
  fi
  if (( i % 2 == 0 )); then
    echo "Skipping even $i"; continue
  fi
  echo "Processing $i"
done
```

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ vim bro7.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ bash bro7.sh
Processing 1
Skipping even 2
Processing 3
Skipping even 4
Reached 5, breaking
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$
```

Exercise 7: Nested loops

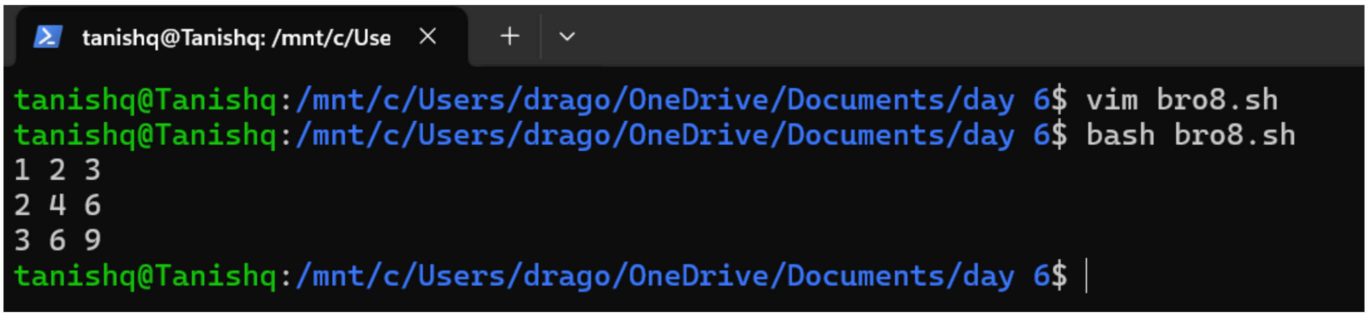
Task Statement:

Create nested loops to generate a multiplication table.

Command(s):

```
for i in {1..3}; do
  for j in {1..3}; do
    echo -n "$((i*j)) "
  done
  echo
done
```

Output:



```
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ vim bro8.sh
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$ bash bro8.sh
1 2 3
2 4 6
3 6 9
tanishq@Tanishq:/mnt/c/Users/drago/OneDrive/Documents/day 6$
```

Result

- Implemented `for`, `while`, and `until` loops and used loop control statements.
- Practiced reading input, processing files, and nested iteration.

Challenges Faced & Learning Outcomes

- Challenge 1: Handling spaces and special characters when iterating filenames — learned to use quotes and `read -r`.
- Challenge 2: Remembering arithmetic syntax in Bash — used `(( ))` and `expr` where needed.

Learning:

- Loops are powerful for automation in shell scripting. Correct quoting and use of control constructs prevent common bugs.

Conclusion

The lab demonstrated practical loop constructs in Bash for automating repetitive tasks and processing data efficiently.