NORTHERN UNIVERSITY
Knowledge for Innovation and Change

COURSE TITLE: OPERATING SYSTEM LAB WORK
COURSE CODE: CSE 3373

REPORT ON: IMPLEMENTATION OF CONDITIONAL STATEMENTS IN BASH

SUBMITTED TO,
NIZIA NAHYAN
LECTURER, NUB(CSE)

SUBMITTED BY,
ZUBAER AHMED ZIHAD
ID: 41230100877
SECTION: 7A

DATE OF SUBMISSION: 27 April 2025

Lab Report: Implementation of Conditional Statements in Bash

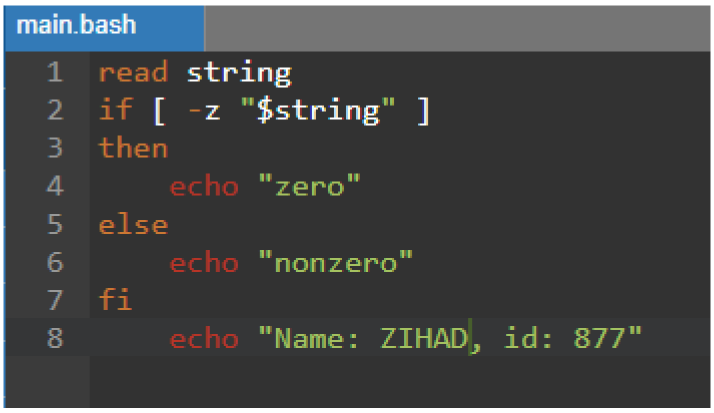
Introduction

In Bash scripting, **conditional statements** allow us to make decisions based on different conditions. Using different operators and structures, we can check if a string is empty, compare two strings, check if a number is even or odd, avoid errors like division by zero, and use **nested if-else** blocks. In this lab, we will learn about important Bash operators and examples step-by-step.

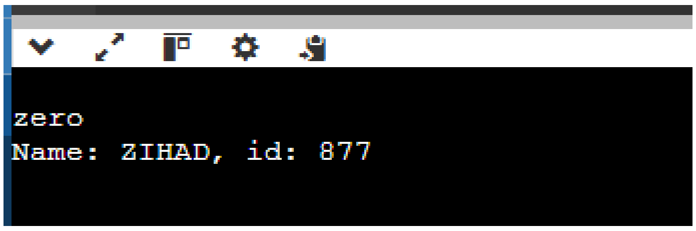
Conditional Statements and Operators

1. -z Operator

- **Function:** Checks if a string is **empty** (zero length).
- **Description:**
  The `-z` operator returns true if the string is empty.



```
read string
if [ -z "$string" ]
then
    echo "zero"
else
    echo "nonzero"
fi
    echo "Name: ZIHAD, id: 877"
```

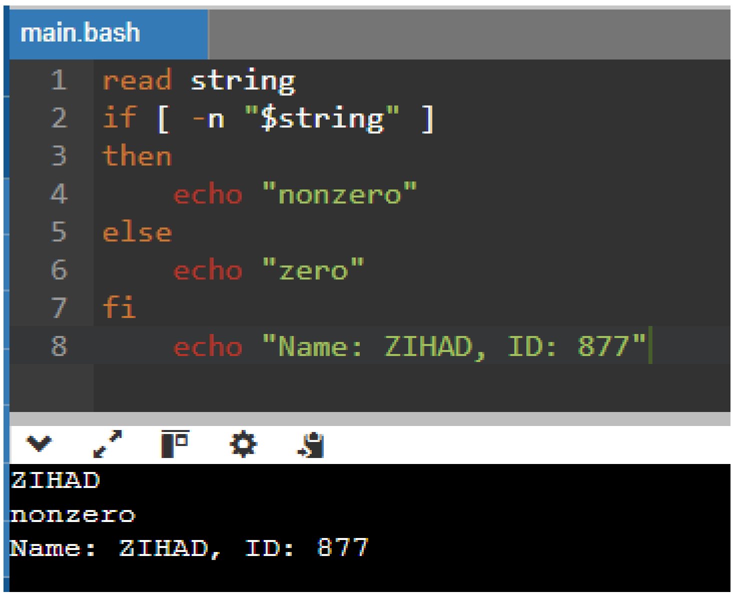


```
zero
Name: ZIHAD, id: 877
```

---

2. -n Operator

- **Function:** Checks if a string is **not empty** (non-zero length).
- **Description:**
  The `-n` operator returns true if the string is not empty.



main.bash

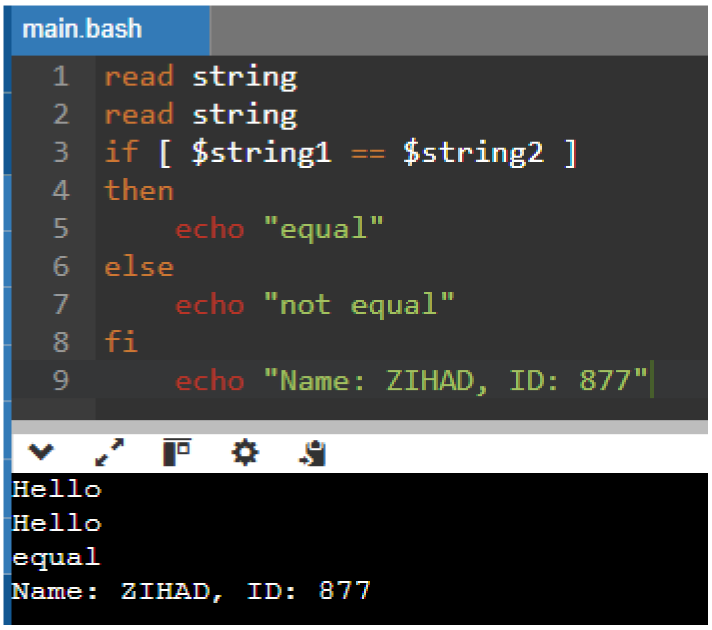
```bash
read string
if [ -n "$string" ]
then
     echo "nonzero"
else
     echo "zero"
fi
     echo "Name: ZIHAD, ID: 877"
```

```
ZIHAD
nonzero
Name: ZIHAD, ID: 877
```

---

3. String Comparison Using == Operator

- **Function:** Compares two strings for **equalit**
- **Description:**
  The == operator checks if two strings are the same.



main.bash

```bash
read string
read string
if [ $string1 == $string2 ]
then
     echo "equal"
else
     echo "not equal"
fi
     echo "Name: ZIHAD, ID: 877"
```

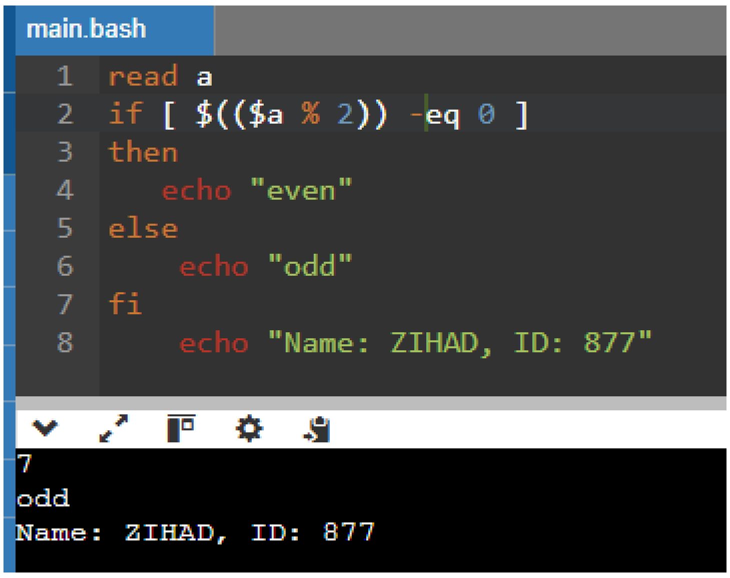
```
Hello
Hello
equal
Name: ZIHAD, ID: 877
```

---

4. Checking Even or Odd (Single Number)

- **Function:** Checks if a number is **even or odd**.

- **Description:**
  We use the modulo operator % to find the remainder when divided by 2.

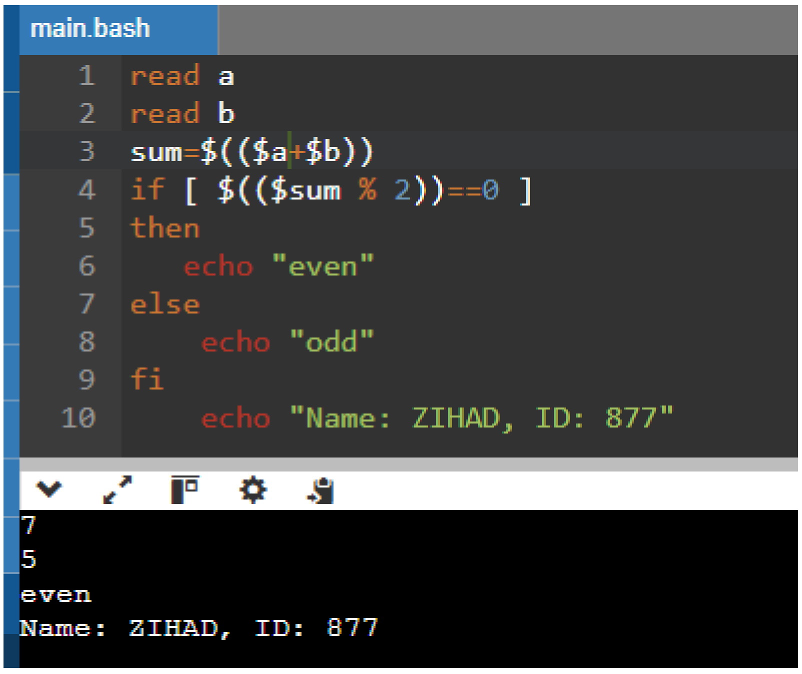


```bash
read a
if [ $(($a % 2)) -eq 0 ]
then
    echo "even"
else
     echo "odd"
fi
     echo "Name: ZIHAD, ID: 877"
```

```
7
odd
Name: ZIHAD, ID: 877
```

5. Sum of Two Numbers and Check Even/Odd

- **Function:** Adds two numbers and checks if the sum is **even or odd**.
- **Description:**
  First, add the numbers, then use modulo to check even or odd.

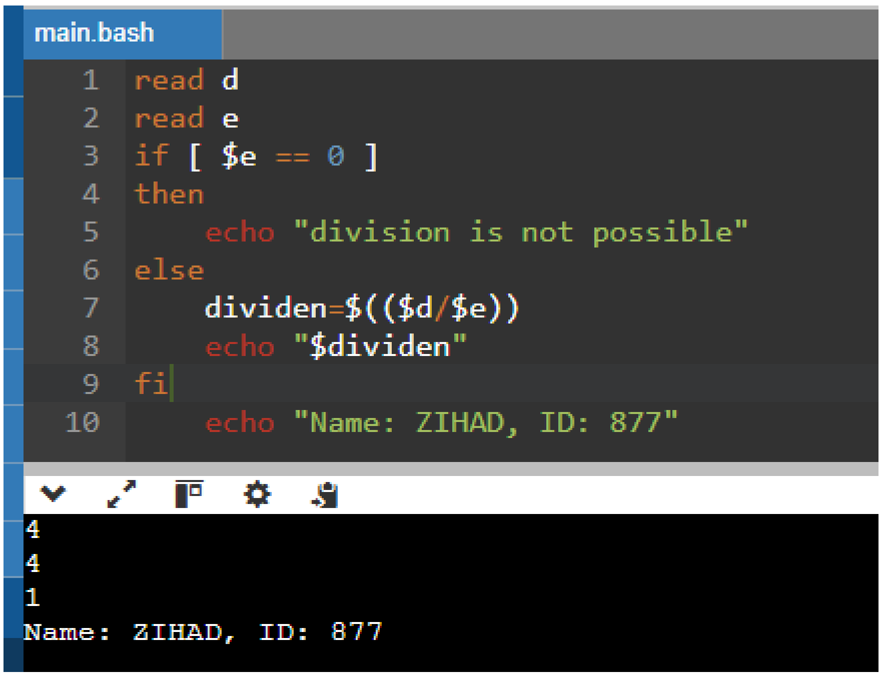


```bash
read a
read b
sum=$(($a+$b))
if [ $(($sum % 2))==0 ]
then
    echo "even"
else
     echo "odd"
fi
     echo "Name: ZIHAD, ID: 877"
```

```
7
5
even
Name: ZIHAD, ID: 877
```

6. Division with Zero Checking

- **Function:** Divides two numbers but first checks if the denominator is **zero**.
- **Description:**
  To avoid errors, we must check if the second number is not zero.


main.bash

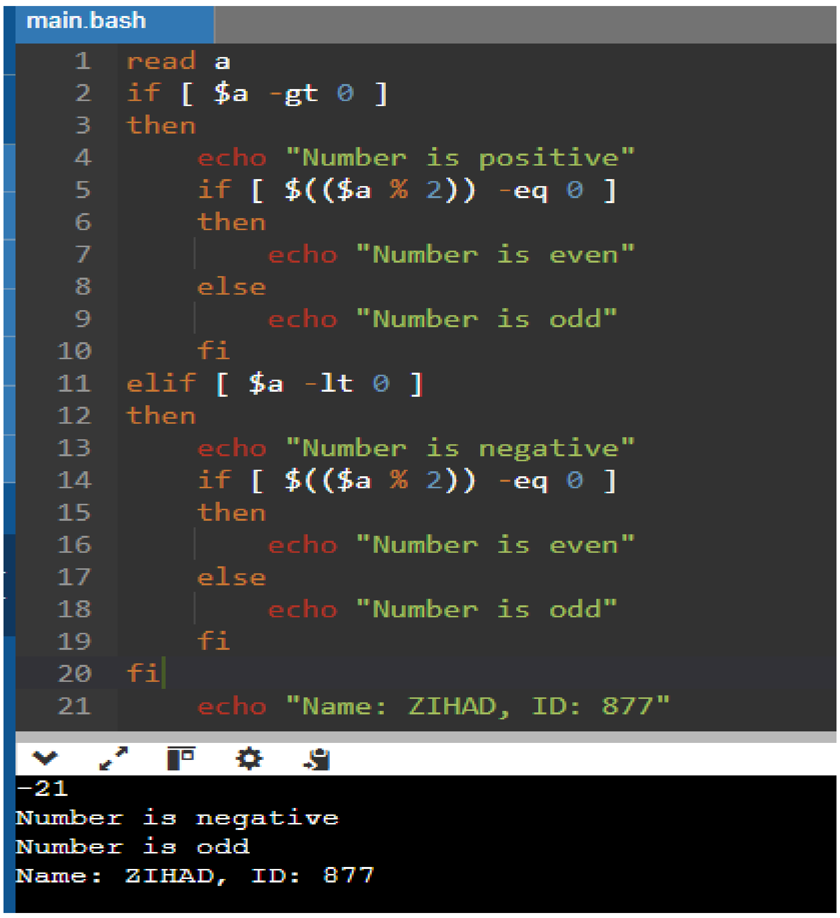
```bash
read d
read e
if [ $e == 0 ]
then
    echo "division is not possible"
else
    dividen=$(($d/$e))
    echo "$dividen"
fi
    echo "Name: ZIHAD, ID: 877"
```

```
4
4
1
Name: ZIHAD, ID: 877
```

---

7. Nested if-else Statement Example

- **Function:** Using **if-else inside another if-else** to check multiple conditions.
- **Description:**
  Nested `if-else` allows checking one condition inside another.


main.bash

```bash
read a
if [ $a -gt 0 ]
then
    echo "Number is positive"
    if [ $(($a % 2)) -eq 0 ]
    then
        echo "Number is even"
    else
        echo "Number is odd"
    fi
elif [ $a -lt 0 ]
then
    echo "Number is negative"
    if [ $(($a % 2)) -eq 0 ]
    then
        echo "Number is even"
    else
        echo "Number is odd"
    fi
fi
    echo "Name: ZIHAD, ID: 877"
```

```
-21
Number is negative
Number is odd
Name: ZIHAD, ID: 877
```

Discussion

Conditional statements make Bash scripts smart and interactive.

- We can check if a string is empty or not using `-z` and `-n`.
- Comparing two strings helps in making decisions based on text values.
- Checking even or odd numbers is useful for numerical logic.
- Checking for division by zero helps avoid errors in calculations.
- Nested if-else statements are very useful for multiple-step decisions.

These features are very important in real-world Bash scripts where automation needs careful checking of inputs and outputs.

Conclusion

In this lab, we explored many useful conditional techniques in Bash scripting.
We learned about checking strings, comparing strings, checking even/odd numbers, handling division by zero, and writing nested if-else blocks.
Practicing these basics will help you build more powerful and error-free Bash scripts for daily tasks or professional automation projects.