# Linux Practice

18 February 2025 12:30

Summary of linux imp commands:

rmdir – remove empty directories
rmdir -p  foldername

rm -rf -> for non-empty directories

mv -> cut paste

uname - displays information about the system's operating system, kernel, and hardware

cat -> print content in file

ps aux -> what is running, how much memory is taking

**Sorting Files**
sort filename -> alphabetical order
sort -r filename -> in reverse order
sort -n filename -> to sort numbers

**Calendar Commands**
sudo apt install ncal -> to install calendar
ncal -> to view calendar

**File & Command Location**
plocate -> to locate files recursively, very fast
whereis -> used to find the location of the source, binary, and manual sections of a command

**Word Counting**
wc -> word counting
Options:
- -c -> count bytes
- -l -> count lines
- -w -> count words

**Searching in Files**
grep -> to search for a pattern in a file

Here is the improved version with proper formatting:

**Process Monitoring**
- top – Provides a real-time view of all running processes.

**Disk Usage Commands**
- du -sh * – Displays the total size of each file and directory in the current directory in a human-readable format.
- du -sh . – Displays the total disk usage of the current directory, including all its subdirectories and files.

**File Compression (Tar & Gzip)**
- tar -zvf etc12.tar.gz – Creates a compressed archive using tar.
  - -z → Compresses the archive using gzip.
  - -v → Enables verbose mode to show file processing.
  - -f etc12.tar.gz → Specifies the filename of the archive.

**Aliases**
- alias c="clear" – Creates an alias for the clear command.

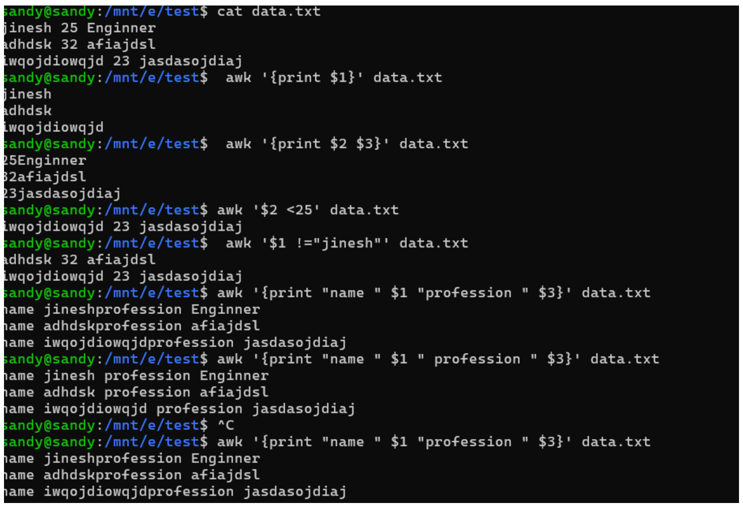
**Network Information**
- ip a – Displays IPv4 and IPv6 addresses.

**Checking Shell**
- echo $SHELL – Shows the current shell being used. Bash is the default shell

**AWK - Pattern Matching and Processing**
- awk is a powerful text-processing tool used for pattern matching, searching, and manipulating text in files.

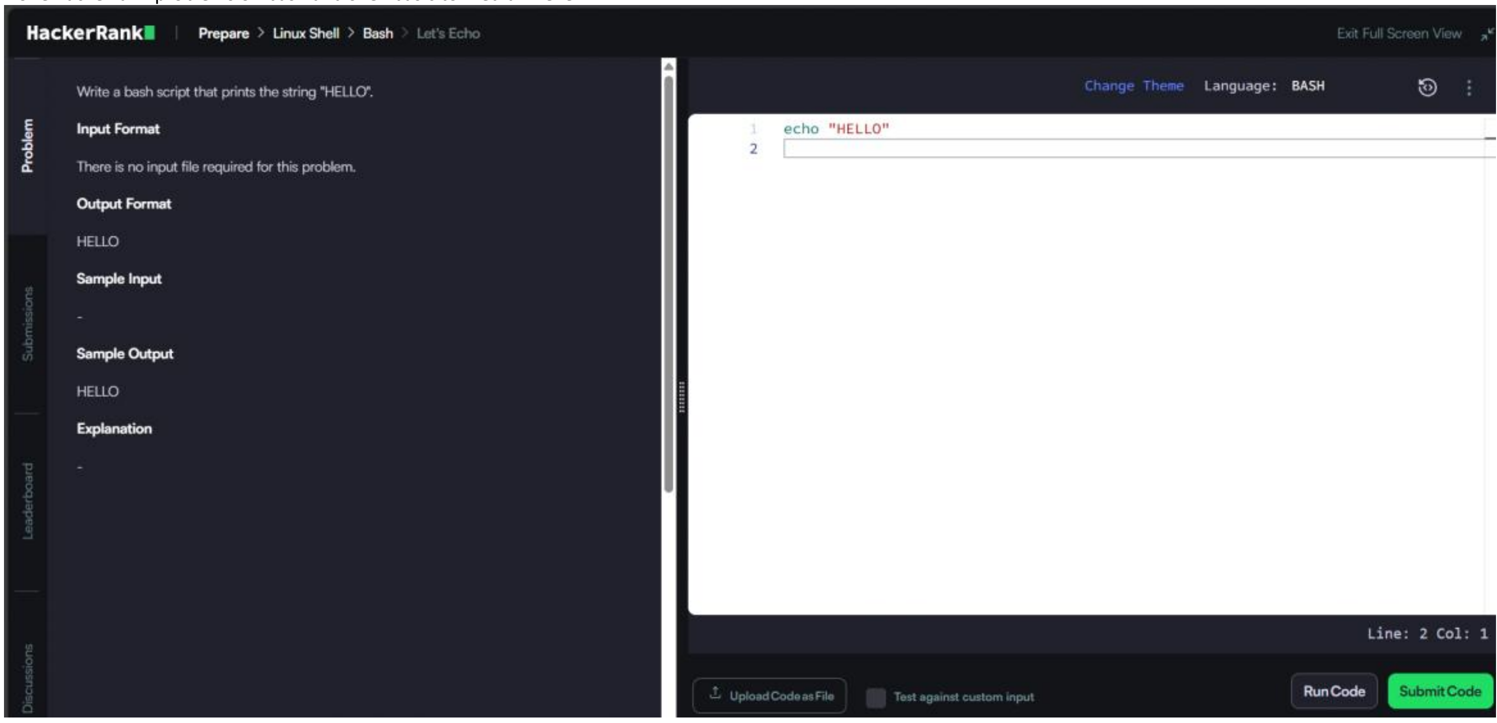
**Basic AWK Commands**
- awk '{print $1}' file.txt → Prints the first column from the file.
- awk -F ":" '{print $1, $3}' /etc/passwd → Uses : as a delimiter and prints the first and third fields from /etc/passwd.
- awk '$3 > 1000 {print $1, $3}' file.txt → Prints the first and third columns where the third column is greater than 1000.
- awk '/error/ {print $0}' log.txt → Searches for lines containing "error" in log.txt and prints them.

**Advanced AWK Usage**
- awk '{sum += $2} END {print "Total:", sum}' file.txt → Sums up the values in the second column and prints the total.
- awk '{if ($3 > 50) print $1, $3}' file.txt → Conditional filtering of data.
- awk '{gsub("old", "new"); print}' file.txt → Replaces all occurrences of "old" with "new" in file.txt.
- awk 'BEGIN {print "Header"} {print $1} END {print "Footer"}' file.txt → Adds a header and footer to the output



```
sandy@sandy:/mnt/e/test$ cat data.txt
jinesh 25 Enginner
adhdsk 32 afiajdsl
iwqojdiowqjd 23 jasdasojdiaj
sandy@sandy:/mnt/e/test$  awk '{print $1}' data.txt
jinesh
adhdsk
iwqojdiowqjd
sandy@sandy:/mnt/e/test$  awk '{print $2 $3}' data.txt
25Enginner
32afiajdsl
23jasdasojdiaj
sandy@sandy:/mnt/e/test$ awk '$2 <25' data.txt
iwqojdiowqjd 23 jasdasojdiaj
sandy@sandy:/mnt/e/test$  awk '$1 !="jinesh"' data.txt
adhdsk 32 afiajdsl
iwqojdiowqjd 23 jasdasojdiaj
sandy@sandy:/mnt/e/test$ awk '{print "name " $1 "profession " $3}' data.txt
name jineshprofession Enginner
name adhdskprofession afiajdsl
name iwqojdiowqjdprofession jasdasojdiaj
sandy@sandy:/mnt/e/test$ awk '{print "name " $1 " profession " $3}' data.txt
name jinesh profession Enginner
name adhdsk profession afiajdsl
name iwqojdiowqjd profession jasdasojdiaj
sandy@sandy:/mnt/e/test$ ^C
sandy@sandy:/mnt/e/test$ awk '{print "name " $1 "profession " $3}' data.txt
name jineshprofession Enginner
name adhdskprofession afiajdsl
name iwqojdiowqjdprofession jasdasojdiaj
```

Done hackerrank problems on bash and shell basic to medium level



Write a bash script that prints the string "HELLO".

**Input Format**

There is no input file required for this problem.

**Output Format**

HELLO

**Sample Input**

-

**Sample Output**

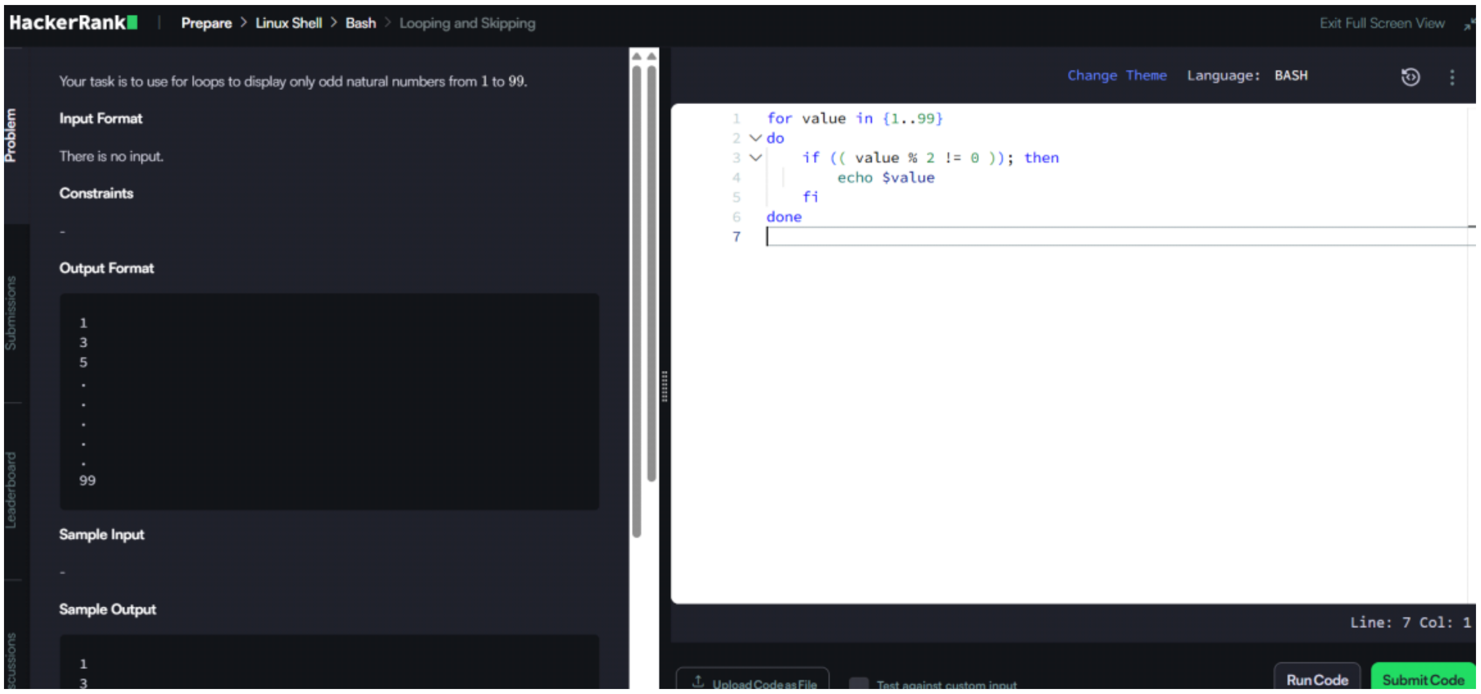
HELLO

**Explanation**

-

```
echo "HELLO"
```

Line: 2 Col: 1



Your task is to use for loops to display only odd natural numbers from 1 to 99.

**Input Format**

There is no input.

**Constraints**

-

**Output Format**

```
1
3
5
.
.
.
.
99
```

**Sample Input**

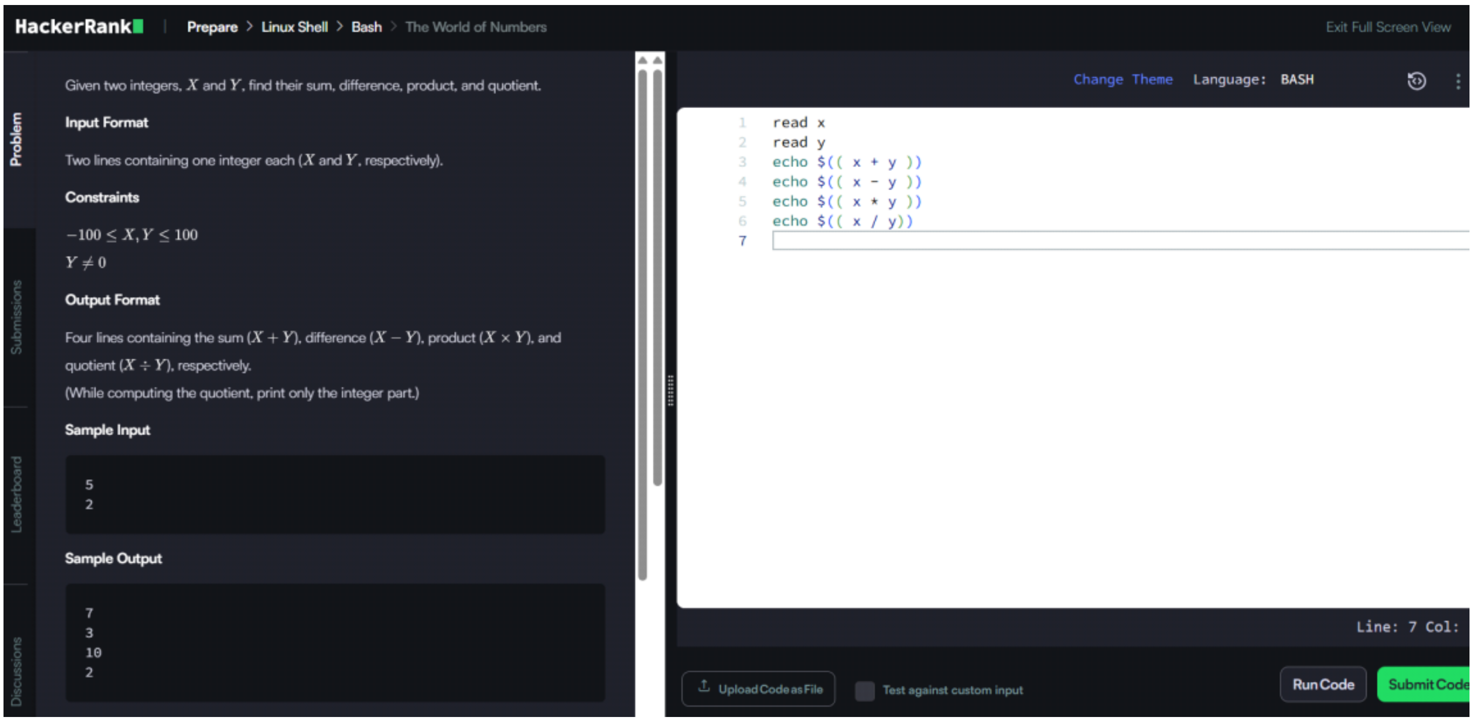
-

**Sample Output**

```
1
3
```

```
for value in {1..99}
do
    if (( value % 2 != 0 )); then
        echo $value
    fi
done
```

Line: 7 Col: 1



Given two integers, $X$ and $Y$, find their sum, difference, product, and quotient.

**Input Format**

Two lines containing one integer each ($X$ and $Y$, respectively).

**Constraints**

$-100 \leq X, Y \leq 100$

$Y \neq 0$

**Output Format**

Four lines containing the sum ($X + Y$), difference ($X - Y$), product ($X \times Y$), and quotient ($X \div Y$), respectively.

(While computing the quotient, print only the integer part.)

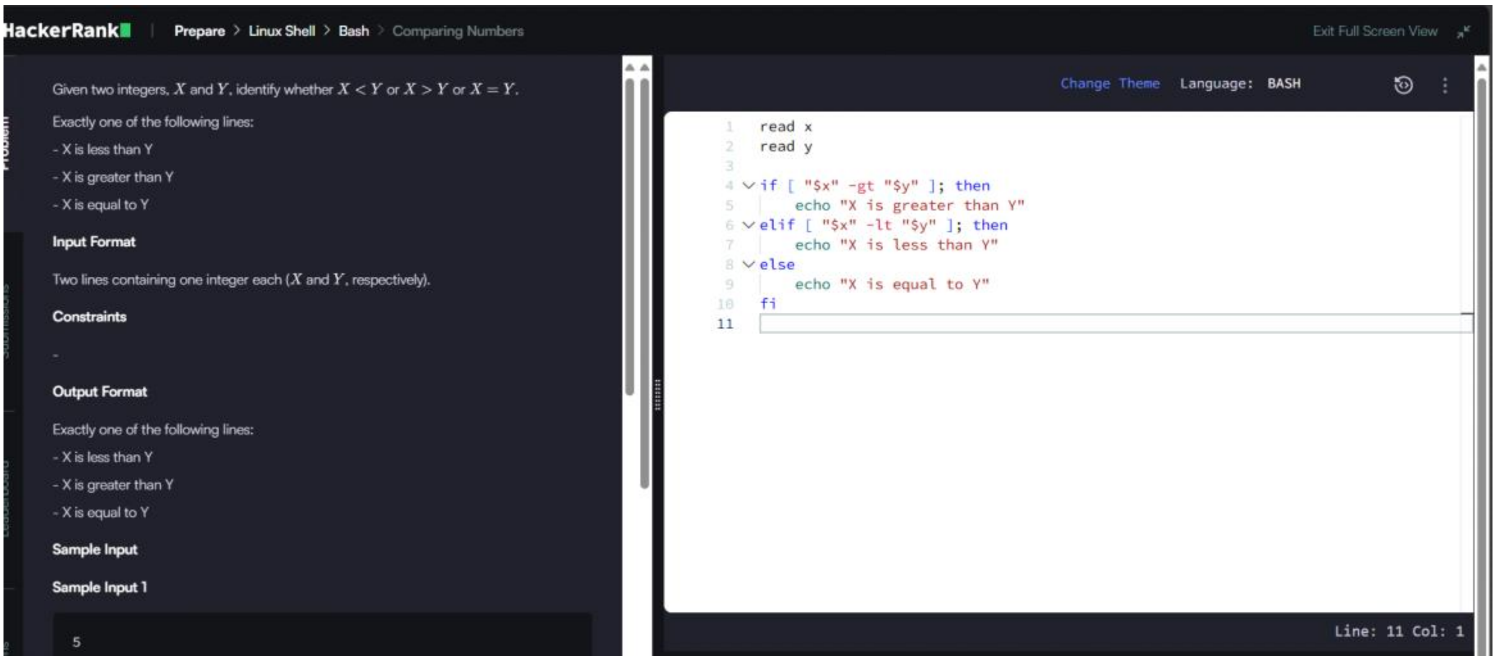
**Sample Input**

```
5
2
```

**Sample Output**

```
7
3
10
2
```

```bash
read x
read y
echo $(( x + y ))
echo $(( x - y ))
echo $(( x * y ))
echo $(( x / y))
```



Given two integers, $X$ and $Y$, identify whether $X < Y$ or $X > Y$ or $X = Y$.

Exactly one of the following lines:

- X is less than Y
- X is greater than Y
- X is equal to Y

**Input Format**

Two lines containing one integer each ($X$ and $Y$, respectively).

**Constraints**

-

**Output Format**

Exactly one of the following lines:

- X is less than Y
- X is greater than Y
- X is equal to Y

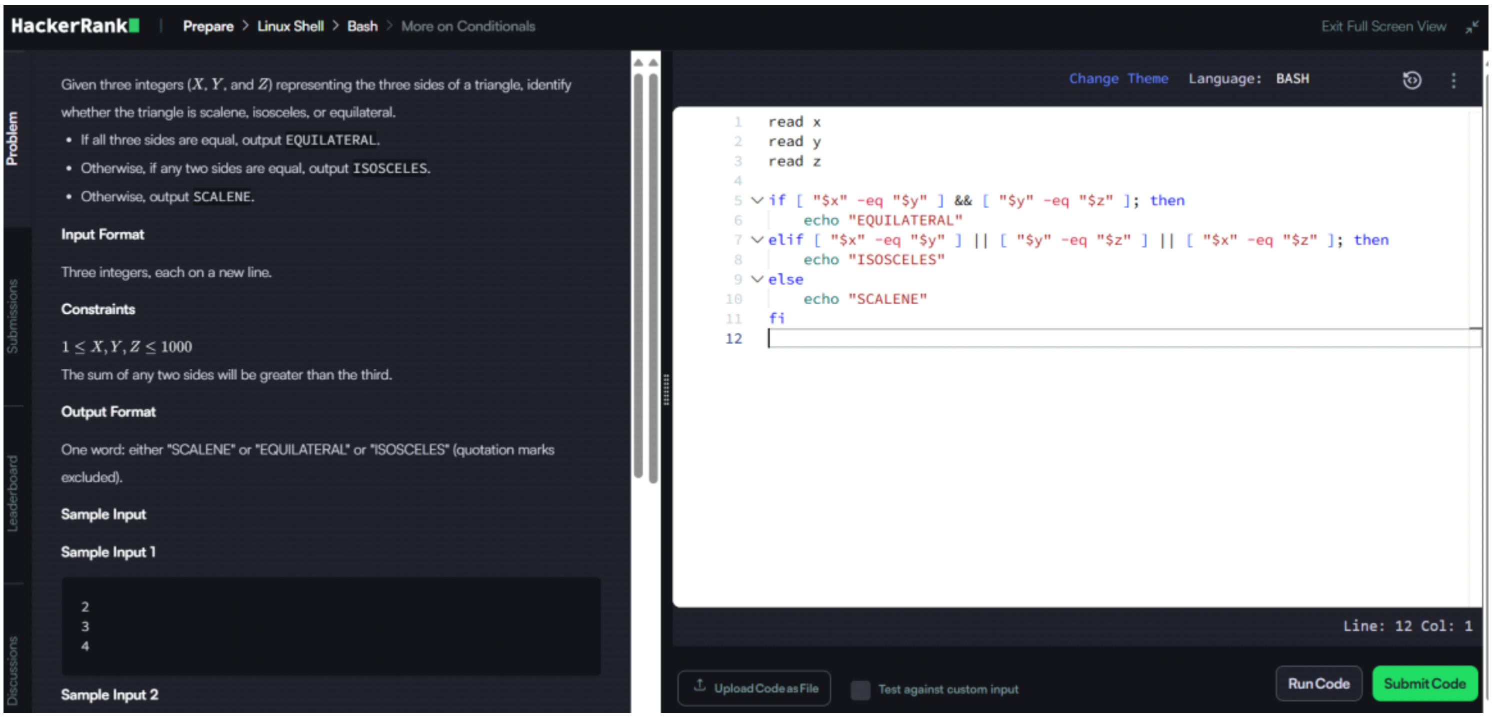
**Sample Input**

**Sample Input 1**

```
5
```

```bash
read x
read y

if [ "$x" -gt "$y" ]; then
    echo "X is greater than Y"
elif [ "$x" -lt "$y" ]; then
    echo "X is less than Y"
else
    echo "X is equal to Y"
fi
```



Given three integers ($X$, $Y$, and $Z$) representing the three sides of a triangle, identify whether the triangle is scalene, isosceles, or equilateral.

- If all three sides are equal, output `EQUILATERAL`.
- Otherwise, if any two sides are equal, output `ISOSCELES`.
- Otherwise, output `SCALENE`.

**Input Format**

Three integers, each on a new line.

**Constraints**

$1 \leq X, Y, Z \leq 1000$

The sum of any two sides will be greater than the third.

**Output Format**

One word: either "SCALENE" or "EQUILATERAL" or "ISOSCELES" (quotation marks excluded).

**Sample Input**

**Sample Input 1**

```
2
3
4
```

**Sample Input 2**

```bash
read x
read y
read z

if [ "$x" -eq "$y" ] && [ "$y" -eq "$z" ]; then
    echo "EQUILATERAL"
elif [ "$x" -eq "$y" ] || [ "$y" -eq "$z" ] || [ "$x" -eq "$z" ]; then
    echo "ISOSCELES"
else
    echo "SCALENE"
fi
```