# Week 5 - Code with us

## Problem 1

Write a bash script that reads two numbers from the standard input and prints the sum of the numbers. Assume that the input will be numbers and nothing else.

```
read var1
read var2
echo $(($var1+$var2))

# Or use
# expr $var1 + $var2
# Or
#sum=`expr $var1 + $var2`
#echo $sum
```

## Problem 2

Write a bash script that takes two arguments, checks if both the arguments are positive integers then prints their sum; else prints "NOT INTEGERS" to STDERR and exit with exit code 1.

Note: Use the below if else conditional statment if needed

```
if condition; then
  ...
  ...
else
  ...
  ...
fi
```

## Solution

```
if [ `echo $1 | egrep '^[0-9]+$'` ] && [ `echo $2 | egrep '^[0-9]+$'` ]; then
  echo $(($1+$2))
else
  echo "NOT INTEGERS" >&2
  exit 1
fi

# Or can use the below if condition to check for integers
#pat='^[0-9]+$'
#if ! [[ $1 =~ $pat && $2 =~ $pat ]]; then
```

# Problem 3

Write a bash script that takes two arguments, checks if both the arguments are positive integers then prints their sum; else concatenate the string values in both the arguments and prints the combined string.

## Solution

```
if [ `echo $1 | egrep '^[0-9]+$'` ] && [ `echo $2 | egrep '^[0-9]+$'` ]; then
  echo $(($1+$2))
else
  echo $1$2
fi

# Or can use the below if condition to check for integers
#pat='^[0-9]+$'
#if ! [[ $1 =~ $pat && $2 =~ $pat ]]; then
```

# Problem 4

Write a bash script that accepts a file path as an argument and checks if that exists and is readable by current user and prints the output as below.

- Prints "DOES NOT EXIST" on STDERR and return with error code 1 if the file does not exist at the given path.
- Prints "NOT READABLE" on STDERR and return with error code 2 if the file is not readable by current user.
- Prints "WOO HOO" if the file exists and is readable too.

Note: Use the below if elif conditional statment if needed

```
if condition; then
  ...
  ...
elif condition; then
  ...
  ...
else
  ...
  ...
fi
```

## Solution

```
if ! [ -e $1 ]; then
  echo "DOES NOT EXISTS" >&2
  exit 1
elif ! [ -r $1 ]; then
  echo "NOT READABLE" >&2
  exit 2
else
  echo "WOO HOO"
fi
```

# Problem 5

Write a bash script which takes in the value of `n` and prints it in reverse order (For example if input number is 123, the Output will be 321)

## Solution

```
n=$1
counter=0
ans=0
while [ $n -gt 0 ]
do
    counter=$(( $n % 10 ))
    ans=$(( $ans * 10 + $counter ))
    n=$(( $n / 10 ))
done
echo $ans
```

# Problem 6

Write a bash script that accepts an integer say `n` and prints the below pattern till `n` lines.

```
*
**
***
****
*****
```

In the above sample the value of `n` is 5.

## Solution

```
n=$1
i=1
while [ $i -le $n ]; do
  j=1
  while [ $j -le $i ]; do
    #printf "*"
    echo -n "*"
    j=$((j+1))
  done
  echo
  i=$((i+1))
done
```

# Problem 7

Write a bash script which takes in the value of `n` and prints whether the number is prime or not. If n is a prime number, the program must print "Prime" and if not, it must print "Not Prime"

## Solution

```
flag=0
n=$1
for (( i=2; i<=$n; i++ )); do
  if [ $((n%i)) -eq 0 ]; then
    flag=1
  fi
done
if [ $flag -eq 0 ]; then
  echo "Prime"
else
  echo "Not Prime"
fi
```

# Problem 8

`df -h` gives the disk/filesystem usage information. Write a bash script to list all the filesystem mount point names based on their percentage usage divided in 5 categories in the format below.

```
0-50
(names of filesystem one in each line with usage between 0 to 50%)
50-75
(names of filesystem one in each line with usage between 50 to 75%)
75-85
(names of filesystem one in each line with usage between 75 to 85%)
85-95
(names of filesystem one in each line with usage between 85 to 95%)
>95
(names of filesystem one in each line with usage above 95%)
```

In each category print the range in one line followed by the filesystem mount point names. Print the range string even if there are no filesystem with usage in that range. Your script should not print anything else, all other errors and ouput from your script should be redirected to /dev/null.

Filesystem mount point name is the last field in the output of df -h.

The categories are

- 0% to less than 50% usage.
- 50% to less than 75% usage.
- 75% to less than 85% usage.
- 85% to less than 95% usage.
- Equal and above 95% usage.

Hint: Can store the df command output in a file. Then work on the file named `dfOutput.csv` line by line using

```
while read -r line;
do
   echo $line; # To print the line.
   # Write your code to process the line.
done < dfOutput.csv
```

Use ${var:0:-1} to remove the last character of string var.

## Solution with arrays

```
df -h >dfOutput.csv

ar1=()
ar2=()
ar3=()
ar4=()
ar5=()
while read -r line
do
  var=`echo $line | cut -d " " -f 5`
  usage=${var:0:-1}
```

```
  if [[ `echo $usage | egrep '^[0-9]+$'` ]]; then
    if [[ $usage < 50 ]]; then
      ar1+=(`echo $line | cut -d " " -f 6`)
    elif [[ $usage < 75 ]]; then
      ar2+=(`echo $line | cut -d " " -f 6`)
    elif [[ $usage < 85 ]]; then
      ar3+=(`echo $line | cut -d " " -f 6`)
    elif [[ $usage < 95 ]]; then
      ar4+=(`echo $line | cut -d " " -f 6`)
    else
      ar5+=(`echo $line | cut -d " " -f 6`)
    fi
  fi
done < dfOutput.csv

echo '0-50'
printf '%s\n' "${ar1[@]}"
echo '50-75'
printf '%s\n' "${ar2[@]}"
echo '75-85'
printf '%s\n' "${ar3[@]}"
echo '85-95'
printf '%s\n' "${ar4[@]}"
echo '>95'
printf '%s\n' "${ar5[@]}"

rm dfOutput.csv 2>/dev/null
```

## Solution with files

```
df -h >dfOutput.csv

touch range1 range2 range3 range4 range5 2>/dev/null
while read -r line
do
  var=`echo $line | cut -d " " -f 5`
  usage=${var:0:-1}
  if [[ `echo $usage | egrep '^[0-9]+$'` ]]; then
    if [[ $usage < 50 ]]; then
      echo $line | cut -d " " -f 6  >>range1
    elif [[ $usage < 75 ]]; then
      echo $line | cut -d " " -f 6  >>range2
    elif [[ $usage < 85 ]]; then
      echo $line | cut -d " " -f 6  >>range3
    elif [[ $usage < 95 ]]; then
      echo $line | cut -d " " -f 6  >>range4
    else
      echo $line | cut -d " " -f 6  >>range5
```

```
    fi
  fi
done < dfOutput.csv

echo '0-50'
cat range1
echo '50-75'
cat range2
echo '75-85'
cat range3
echo '85-95'
cat range4
echo '>95'
cat range5

rm dfOutput.csv range1 range2 range3 range4 range5 2>/dev/null
```

# Problem 9

In Problem 5, modify the output of your script as below.

- Print the range string only if there is a filesystem in that range.

For example if there is no filesystem with usage >95% and also none in the range 75-85, and rest all range has atleast one filesystem with usage in that range than your output should be

```
0-50

(names of filesystem one in each line with usage between 0 to 50%)

50-75

(names of filesystem one in each line with usage between 50 to 75%)

85-95

(names of filesystem one in each line with usage between 85 to 95%)
```

## Solution

```
df -h >dfOutput.csv

touch range1 range2 range3 range4 range5 2>/dev/null
while read -r line
do
  var=`echo $line | cut -d " " -f 5`
  usage=${var:0:-1}
  if [[ `echo $usage | egrep '^[0-9]+$'` ]]; then
```

```
    if [[ $usage < 50 ]]; then
      echo $line | cut -d " " -f 6  >>range1
    elif [[ $usage < 75 ]]; then
      echo $line | cut -d " " -f 6  >>range2
    elif [[ $usage < 85 ]]; then
      echo $line | cut -d " " -f 6  >>range3
    elif [[ $usage < 95 ]]; then
      echo $line | cut -d " " -f 6  >>range4
    else
      echo $line >&2
      echo $line | cut -d " " -f 6  >>range5
    fi
  fi
done < dfOutput.csv

if [[ -s range1 ]]; then
  echo '0-50'
  cat range1
fi
if [[ -s range2 ]]; then
  echo '50-75'
  cat range2
fi
if [[ -s range3 ]]; then
  echo '75-85'
  cat range3
fi
if [[ -s range4 ]]; then
  echo '85-95'
  cat range4
fi
if [[ -s range5 ]]; then
  echo '>95'
  cat range5
fi

rm dfOutput.csv range1 range2 range3 range4 range5 2>/dev/null
```