Array/SelectionSort.sh

Program To Perform Selection Sort

```
# Program To Perform Selection Sort

selectionSort() {
    ar=($*)
    for ((i = 0; i < $(($# - 1)); i++))
    do
        minIndex=$i
    done

```

Array/String/Compare.sh

Program to check whether strings passed are equal or not

```
# Program to check whether strings passed are equal or not

function_compare() {
    string1="$1"
    string2="$2"
    len1=${#string1}
    len2=${#string2}
    i=0
    flag=0
    isEqual=1
    while [ $flag -eq 0 ]
    do
        if [ $i -lt $len1 ]
        then
            char1=${string1:$i:1}
            ascii1=`printf "%d" "'$char1"`
        else
            ascii1=0
            flag=1
        fi
        if [ $i -lt $len2 ]
        then
            char2=${string2:$i:1}
            ascii2=`printf "%d" "'$char2"`
        else
            ascii2=0
            flag=1
        fi

        diff=$(($ascii2 - $ascii1))
        if (($diff != 0))
        then
            isEqual=0
            flag=1
            break
        fi
        i=$(($i + 1))
    done
    if (( $isEqual == 1 ))
    then
        echo "The strings passed equal"
```

```
    else
        echo "The strings are not equal"
    fi
}
```

Function/Combinations.sh

Program To Compute Combinations

```
# Program To Compute Combinations
# nCr = n!/ [r!  * (n - r)!]

factorial() {
    if [ $1 -eq 0 ]
    then
        echo "1"
    else
        echo $(($1 * $(factorial $(($1 - 1)))))
    fi
}

permutation() {
    echo $(( $(factorial $1) / $(factorial $(($1 - $2))) ))
}

combination() {
    echo $(( $(permutation $1 $2) / $(factorial $2) ))
}
```

Function/EuclideanGCD.sh

Function To Evaluate GCD Of Two Numbers Using Recursive Euclidean Algorithm

```
# Function To Evaluate GCD Of Two Numbers Using Recursive Euclidean Algorithm

gcd() {
    if [ $1 -eq 0 ]; then echo $2
    else echo $(gcd $(($2 % $1)) $1); fi
}
```

Function/FactorialLoop.sh

Function To Compute Factorial Of Number With Loop

```
# Function To Compute Factorial Of Number With Loop

factorial() {
    number=$1
    factorial=1

    for ((i = 1; i <= $number; i++))
    do
        factorial=$(($factorial * $i))
    done

```

```
    echo $factorial
}
```

Function/FactorialRecursion.sh

Function To Compute Factorial Of A Number Using Recursion

```
# Function To Compute Factorial Of A Number Using Recursion

factorial() {
    if [ $1 -eq 0 ]
    then
        echo "1"
    else
        echo $(($1 * $(factorial $(($1 - 1)))))
    fi
}
```

Function/Fibonacci.sh

Function To Print Nth Term Of Fibonacci Sequence Using Recursion

```
# Function To Print Nth Term Of Fibonacci Sequence Using Recursion

fibonacci() {
    if [ $1 -eq 0 ]; then echo "0"
    elif [ $1 -eq 1 ]; then echo "1"
    else echo $(( $(fibonacci $(($1 - 1))) + $(fibonacci $(($1 - 2))) ))
    fi
}
```

Function/NthPower.sh

Function To Compute Nth Power Of A Number Using Recursion

```
# Function To Compute Nth Power Of A Number Using Recursion

power() {
    if [ $2 -eq 0 ]; then echo "1"
    else echo $(($1 * $(power $1 $(($2 - 1)))))
    fi
}
```

Function/PascalTriangle.sh

Function To Print Pascal's Trianle N Rows

```
# Function To Print Pascal's Trianle N Rows

factorial() {
    if [ $1 -eq 0 ]
```

```
    then
        echo "1"
    else
        echo $(($1 * $(factorial $(($1 - 1)))))
    fi
}

permutation() {
    echo $(( $(factorial $1) / $(factorial $(($1 - $2))) ))
}

combination() {
    echo $(( $(permutation $1 $2) / $(factorial $2) ))
}

pascal() {
    for ((i = 0; i < $1; i++))
    do
        for ((j = 0; j < $(($1 - $i)); j++))
        do
            printf " "
        done
        for ((k = 0; k <= i; k++))
        do
            printf "%3d" $(combination $i $k)
        done
        echo
    done
}
```

Function/Permutation.sh

Program To Compute Permutaions

```
# Program To Compute Permutaions
# nPr = n!/(n - r)!

factorial() {
    if [ $1 -eq 0 ]
    then
        echo "1"
    else
        echo $(($1 * $(factorial $(($1 - 1)))))
    fi
}

permutation() {
    echo $(( $(factorial $1) / $(factorial $(($1 - $2))) ))
}
```

Function/Upper.sh

Conversion Of String From Lower Case To Upper Case Using Transliteration

```
# Conversion Of String From Lower Case To Upper Case Using Transliteration
upperCase() {
    echo "Upper Case: `echo $1 | tr [:lower:] [:upper:]:`"
}
```

IfElse/EvenOdd.sh

Program To Check Whether A Number Is Even Or Odd

```
# Program To Check Whether A Number Is Even Or Odd

printf "Enter A Number: "
read number
if [ `expr $number % 2` -eq 0 ]
then
    echo "Even Number"
else
    echo "Odd Number"
fi
---
```

IfElse/Grade.sh

Program to input marks of five subjects Physics, Chemistry, Biology, Mathematics and Computer.

```
# Program to input marks of five subjects Physics, Chemistry, Biology, Mathematics and Computer.
# Calculate percentage and grade according to following:
# Percentage >= 90% : Grade A
# Percentage >= 80% : Grade B
# Percentage >= 70% : Grade C
# Percentage >= 60% : Grade D
# Percentage >= 40% : Grade E
# Percentage < 40% : Grade F
read -p "Enter Marks In Physics: " physics
read -p "Enter Marks In Chemistry: " chemistry
read -p "Enter Marks In Biology: " biology
read -p "Enter Marks In Math: " math
read -p "Enter Marks In Computer: " computer

percentage=$(( ($physics + $chemistry + $biology + $math + $computer) / 5 ))

echo "Percentage: $percentage%"

if (( $percentage >= 90 ))
then
    echo "Grade A"
elif (( $percentage >= 80 ))
then
    echo "Grade B"
elif (( $percentage >= 70 ))
then
    echo "Grade C"
elif (( $percentage >= 60 ))
then
    echo "Grade D"
elif (( $percentage >= 40 ))
then
    echo "Grade E"
else
    echo "Grade F"
fi
```

IfElse/Greater3.sh

Program To Find Greatest Of 3 Numbers

```
# Program To Find Greatest Of 3 Numbers

printf "Enter A : "
read A
printf "Enter B : "
read B
printf "Enter C : "
read C

if [ $A -gt $B ]
then
    if [ $A -gt $C ]
    then
        echo "A Is Greatest"
    else
        echo "C Is Greatest"
    fi
elif [ $B -gt $C ]
then
    echo "B Is Greatest"
else
    echo "C Is Greatest"
fi
```

IfElse/LeapYear.sh

Program To Check Whether A Year Is A Leap Year

```
# Program To Check Whether A Year Is A Leap Year

printf "Enter Year: "
read year
if [ `expr $year % 4` -eq 0 ]
then
    echo "Leap Year"
else
    echo "Not A Leap  Year"
fi
```

IfElse/LUcase.sh

Program To Check Whether The Alphabet Is Lowercase Or Uppercase Using Regex

```
# Program To Check Whether The Alphabet Is Lowercase Or Uppercase Using Regex
read -p "Enter An Alphabet: " -n1 alpha
echo

if [[ $alpha =~ [[:upper:]] ]]
then
    echo "Uppercase Alphabet"
elif [[ $alpha =~ [[:lower:]] ]]
then
    echo "Lowercase Alphabet"
else
    echo "Not An Alphabet"
fi
```

IfElse/PositiveNegative.sh

Program To Check Whether A Number A Number Is Positive, Negative Or Zero

```
# Program To Check Whether A Number A Number Is Positive, Negative Or Zero

printf "Enter A Number: "
read number

if [ $number -gt 0 ]
then
    echo "Positive Number"
elif [ $number -lt 0 ]
then
    echo "Negative Number"
else
    echo "Zero"
fi
```

IfElse/ProfitLoss.sh

Program To Calculate Profit Or Loss

```
# Program To Calculate Profit Or Loss

read -p "Enter Cost Price: " cp
read -p "Enter Selling Price: " sp

delta=$(($sp - $cp))

if [ $delta -gt 0 ]
then
    echo "Profit: Rs.$delta"
    ppercent=`echo "scale=3; $delta / $cp * 100" | bc`
    echo "Profit Percentage: $ppercent%"
elif [ $delta -lt 0 ]
then
    echo "Loss: Rs.$delta"
    lpercent=`echo "scale=3; $delta / $cp * 100" | bc`
    echo "Loss Percentage: $lpercent%"
else
    echo "Neither Profit Nor Loss"
fi
```

IfElse/Quadrant.sh

Program To Accept A Coordinate Point (X, Y) And Determine Which Quadrant The Point Lies

```
# Program To Accept A Coordinate Point (X, Y) And Determine Which Quadrant The Point Lies

read -p "Enter X Co-ordinate: " x
read -p "Enter Y Co-ordinate: " y

if (( x > 0 && y > 0 ))
then
    echo "First Quadrant"
elif (( x < 0 && y > 0 ))
```

```
then
    echo "Second Quadrant"
elif (( x < 0 && y < 0 ))
then
    echo "Third Quadrant"
elif (( x > 0 && y < 0 ))
then
    echo "Fourth Quadrant"
else
    echo "Point Lies On Axes"
fi
```

IfElse/Roots.sh

Program To Calculate The Nature Of Roots Of A Quadratic Equation

```
# Program To Calculate The Nature Of Roots Of A Quadratic Equation

read -p "Enter The Value Of a: " a
read -p "Enter The Value Of b: " b
read -p "Enter The Value Of c: " c

discriminant=$(($b * $b - $((4 * $a * $c))))

printf "Discriminant: %d\n" $discriminant

if [ $discriminant -eq 0 ]
then
    echo "Roots Are Real And Equal"
elif [ $discriminant -gt 0 ]
then
    echo "Roots Are Real And Distinct"
else
    echo "Roots Are Imaginary"
fi
```

IfElse/Triangle.sh

Program To Determine Type Of Triangle

```
# Program To Determine Type Of Triangle

printf "Enter side A : "
read A
printf "Enter side B : "
read B
printf "Enter side C : "
read C

if [ $A -le 0 -o $B -le 0 -o $C -le 0 ]
then
    echo "Invalid Triangle"
elif [ $(($A + $B)) -le $C -o $(($B + $C)) -le $A -o $(($A + $C)) -le $B ]
then
    echo "Invalid Triangle"
elif [ $A -eq $B -a $B -eq $C ]
then
    echo "Equilateral Triangle"
elif [ $A -eq $B -o $B -eq $C -o $C -eq $A ]
then
    echo "Isoceles Triangle"
```

```
else
    echo "Scalene Triangle"
fi
```

IfElse/Vowel.sh

Program To Check Whether The Alphabet Is Vowel Or Consonant

```
# Program To Check Whether The Alphabet Is Vowel Or Consonant

read -p "Enter An Alphabet: " -n1 alpha
echo

if [[ $alpha =~ [aeiouAEIOU] ]]
then
    echo "Vowel"
elif [[ $alpha =~ [bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ] ]]
then
    echo "Consonant"
else
    echo "Not An Alphabet"
fi
```

Loop/APseries.sh

Program To Print An Arithmetic Progression Upto N Terms

```
# Program To Print An Arithmetic Progression Upto N Terms

read -p "Enter The Value Of First Term (a): " a
read -p "Enter The Value Of Common Difference (d): " d
read -p "Enter N: " n

for ((i = 1; i <= n; i++))
do
    an=$(($a + ($i - 1) * $d))
    printf "%d " $an
done
echo
```

Loop/Armstrong.sh

Program To Check Whether A Number Is Armstrong Or Not

```
# Program To Check Whether A Number Is Armstrong Or Not

read -p "Enter A Number: " number
temp=$number
sum=0

while [ $number -gt 0 ]
do
    digit=$(($number % 10))
    number=$(($number / 10))
    sum=$(($sum + $digit * $digit * $digit))
done
```

```
13
14 if [ $sum -eq $temp ]
15 then
16     echo "Armstrong Number"
17 else
18     echo "Not An Armstrong Number"
19 fi
```

Loop/Average.sh

Program To Input N Numbers And Calculate Their Average

```
1  # Program To Input N Numbers And Calculate Their Average
2
3  flag=1
4  length=0
5  sum=0
6
7  while [ $flag -eq 1 ]
8  do
9      read -p "Enter A Number: " input
10     sum=$(($sum + $input))
11     length=$(($length + 1))
12     read -n1 -p "Press 1 To Enter More Or Any Other Key To Exit: " flag
13     echo
14 done
15
16 average=`echo "scale=4; $sum / $length" | bc`
17 echo "Average: $average"
```

Loop/BinaryToDecimal.sh

Program To Convert Binary To Decimal

```
1  # Program To Convert Binary To Decimal
2
3  read -p "Enter A Binary Number: " binary
4
5  len=${#binary}
6  j=1
7  result=0
8
9  for ((i = len - 1; i >= 0; i--))
10 do
11     digit=${binary:i:1}
12     result=$(($result + $j * $digit))
13     j=$(($j * 2))
14 done
15
16 echo "Decimal: $result"
```

Loop/BinaryToHex.sh

Program To Convert Binary To Hexadecimal

```
1  # Program To Convert Binary To Hexadecimal
```

```
2
3   read -p "Enter A Binary Number: " binary
4   len=${#binary}
5   j=1
6   temp=0
7   result=""
8
9   for ((i = len - 1; i >= 0; i--))
10  do
11      digit=${binary:i:1}
12      temp=$(($temp + $digit * $j))
13      j=$(($j * 2))
14      if (($j == 16 || i == 0))
15      then
16          if (($temp < 10)); then result="$temp$result";
17          elif (($temp == 10)); then result="A$result"
18          elif (($temp == 11)); then result="B$result"
19          elif (($temp == 12)); then result="C$result"
20          elif (($temp == 13)); then result="D$result"
21          elif (($temp == 14)); then result="E$result"
22          elif (($temp == 15)); then result="F$result"; fi
23          j=1
24          temp=0
25      fi
26  done
27
28  echo "Hexadecimal: $result"
```

Loop/BinaryToOctal.sh

Program To Convert Binary To Hexadecimal

```
1   # Program To Convert Binary To Hexadecimal
2
3   read -p "Enter A Binary Number: " binary
4   len=${#binary}
5   j=1
6   temp=0
7   result=""
8
9   for ((i = len - 1; i >= 0; i--))
10  do
11      digit=${binary:i:1}
12      temp=$(($temp + $digit * $j))
13      j=$(($j * 2))
14      if (($j == 8 || i == 0))
15      then
16          result="$temp$result";
17          j=1
18          temp=0
19      fi
20  done
21
22  echo "Octal: $result"
```

Loop/Circle.sh

Program To Input Radius Of A Circle And Print It's Circumference And Area

```
1   # Program To Input Radius Of A Circle And Print It's Circumference And Area
2
```

```
read -p "Enter Radius: " radius

circumference=`echo "scale=5; 2 * 3.14159 * $radius" | bc`
area=`echo "scale=5; 3.14159 * $radius * $radius" | bc`

printf "Circumference: %f\nArea: %f\n" $circumference $area
```

Loop/DecimalToBinary.sh

Program To Convert A Decimal Number Into Binary

```
# Program To Convert A Decimal Number Into Binary

read -p "Enter A Number: " number
binary=""

while [ $number -gt 0 ]
do
    digit=$(($number % 2))
    number=$(($number / 2))
    binary="$digit$binary"
done

printf "Binary Equivalent: %s\n" $binary
```

Loop/DecimalToHex.sh

Program To Convert A Decimal Number Into Hexadecimal

```
# Program To Convert A Decimal Number Into Hexadecimal

read -p "Enter A Number: " number
hex=""

while [ $number -gt 0 ]
do
    digit=$(($number % 16))
    number=$(($number / 16))
    if (($digit < 10)); then hex="$digit$hex"
    elif (($digit == 10)); then hex="A$hex"
    elif (($digit == 11)); then hex="B$hex"
    elif (($digit == 12)); then hex="C$hex"
    elif (($digit == 13)); then hex="D$hex"
    elif (($digit == 14)); then hex="E$hex"
    elif (($digit == 15)); then hex="F$hex"; fi
done

printf "Hexadecimal Equivalent: %s\n" $hex
```

Loop/DecimalToOctal.sh

Program To Convert A Decimal Number Into Octal

```
# Program To Convert A Decimal Number Into Octal

read -p "Enter A Number: " number
```

```
oct=""

while [ $number -gt 0 ]
do
    digit=$(($number % 8))
    number=$(($number / 8))
    if (($digit < 10)); then oct="$digit$oct"
    elif (($digit == 10)); then oct="A$oct"
    elif (($digit == 11)); then oct="B$oct"
    elif (($digit == 12)); then oct="C$oct"
    elif (($digit == 13)); then oct="D$oct"
    elif (($digit == 14)); then oct="E$oct"
    elif (($digit == 15)); then oct="F$oct"; fi
done

printf "Octal Equivalent: %s\n" $oct
```

Loop/EuclideanGCD.sh

Program To Calculate GCD Using Euclidean Algorithm

```
# Program To Calculate GCD Using Euclidean Algorithm

read -p "Enter First Number: " a
read -p "Enter Second Number: " b

while ((a != 0))
do
    temp=$a
    a=$(($b % $a))
    b=$temp
done

echo "GCD: $b"
```

Loop/FactorialSeries.sh

Program To Print The Sum Of Following Series Upto N Terms:

```
# Program To Print The Sum Of Following Series Upto N Terms:
# 1! + 2! + 3! + 4! + ....

read -p "Enter N: " n
sum=0

for ((i = 1; i <= $n; i++))
do
    factorial=1
    for ((j = 1; j <= $i; j++))
    do
        factorial=$(($factorial * $j))
    done
    sum=$(($sum + $factorial))
done

echo "Sum: $sum"
```

Loop/Factorial.sh

Program To Find Factorial Of A Number

```
# Program To Find Factorial Of A Number

read -p "Enter A Number: " number
factorial=1

for ((i = 1; i <= $number; i++))
do
    factorial=$(($factorial * $i))
done

printf "Factorial: "
echo $factorial
```

Loop/Factors.sh

Prorgam To Find Factors Of A Nuumber

```
# Prorgam To Find Factors Of A Nuumber

read -p "Enter A Number: " number

printf "Factors: "
for ((i = 1; i <= $(($number / 2 + 1)); i++))
do
    if (($number % $i == 0))
    then
        printf "%d " $i
    fi
done
echo $number
```

Loop/Fibonacci.sh

Program To Print Fibonacci Series Upto N

```
# Program To Print Fibonacci Series Upto N

read -p "Enter N: " n
a=0
b=1
if [ $n -eq 1 ]; then echo "0";
elif [ $n -eq 2 ]; then echo "0 1";
elif [ $n -ge 3 ]; then
    printf "0 1 "
    while [ $n -ge 3 ]
    do
        c=$(($a + $b))
        a=$b
        b=$c
        printf "%d " $c
        n=$(($n - 1))
    done
    echo
fi
```

Loop/FloydTriangle.sh

Program To Print Floyd Triangle Upto N Rows

```
# Program To Print Floyd Triangle Upto N Rows

read -p "Enter The Number Of Rows: " rows
for ((i = 0; i < $rows; i++))
do
    c=$((($i % 2) + 1))
    for ((j = 0; j <= $i; j++))
    do
        printf "%d" $(($c % 2))
        c=$(($c + 1))
    done
    echo
done
```

Loop/GPseries.sh

Program To Print An Geometric Progression Upto N Terms

```
# Program To Print An Geometric Progression Upto N Terms

read -p "Enter The Value Of First Term (a): " a
read -p "Enter The Value Of Common Ratio (r): " r
read -p "Enter N: " n

for ((i = 1; i <= n; i++))
do
    rn=1
    for ((j = 1; j < $i; j++))
    do
        rn=$(($rn * $r))
    done
    tn=$(($a * $rn))
    printf "%d " $tn
done
echo
```

Loop/LCM.sh

Program To Calculate LCM Of Two Number

```
# Program To Calculate LCM Of Two Number

read -p "Enter First Number: " a
read -p "Enter Second Number: " b
x=$a
y=$b

while ((a != 0))
do
    temp=$a
    a=$(($b % $a))
    b=$temp
done
```

```
14
15  echo "LCM: $((($x * $y) / $b))"
```

Loop/NthPower.sh

Program To Calculate Nth Power Of A Number

```
1   # Program To Calculate Nth Power Of A Number
2
3   read -p "Enter A Number: " number
4   read -p "Enter N: " n
5
6   result=1
7
8   for ((i = 0; i < $n; i++))
9   do
10      result=$(($result * $number))
11  done
12
13  echo "Result: $result"
```

Loop/PalindromeNumber.sh

Program To Check Whether A Number Palindrome Or Not

```
1   # Program To Check  Whether A Number Palindrome Or Not
2
3   read -p "Enter A Number: " number
4   temp=$number
5   reverse=0
6
7   while [ $number -gt 0 ]
8   do
9       digit=$(($number % 10))
10      number=$(($number / 10))
11      reverse=$(($reverse * 10 + digit))
12  done
13
14  if [ $reverse -eq $temp ]
15  then
16      echo "Palindrome Number"
17  else
18      echo "Not An Palindrome Number"
19  fi
```

Loop/Pattern1.sh

Program To Print Following Pattern Specified Number Of Rows

```
1   # Program To Print Following Pattern Specified Number Of Rows
2   # *
3   # **
4   # ***
5   # ****
6   # *****
7   # ******
```

```
# ...n

read -p "Enter The Number Of Rows: " rows
i=0
while [ $i -lt $rows ]
do
    cols=0
    while [ $cols -le $i ]
    do
        printf "*"
        cols=$(($cols + 1))
    done
    echo
    i=$(($i + 1))
done
```

Loop/Pattern2.sh

Program To Print Following Pattern Specified Number Of Rows

```
# Program To Print Following Pattern Specified Number Of Rows
# *******
# ******
# *****
# ****
# ***
# **
# *
# ...n

read -p "Enter The Number Of Rows: " rows
while [ $rows -gt 0 ]
do
    cols=0
    while [ $cols -lt $rows ]
    do
        printf "*"
        cols=$(($cols + 1))
    done
    echo
    rows=$(($rows - 1))
done
```

Loop/Pattern3.sh

Program To Print Following Pattern Specified Number Of Rows

```
# Program To Print Following Pattern Specified Number Of Rows
#    *
#   ***
#  *****
# *******
# ...n

read -p "Enter The Number Of Rows: " rows
i=1
while [ $rows -gt 0 ]
do
    spaces=$(($rows - 1))
    while [ $spaces -gt 0 ]
    do
```

```
        printf " "
        spaces=$(($spaces - 1))
    done
    cols=$i
    while [ $cols -gt 0 ]
    do
        printf "*"
        cols=$(($cols - 1))
    done
    echo
    i=$(($i + 2))
    rows=$(($rows - 1))
done
```

Loop/Perfect.sh

Program To Check Whether A Number Is Perfect Or Not

```
# Program To Check Whether A Number Is Perfect Or Not
# 6 = 3 X 2 X 1, 6 = 3 + 2 + 1

read -p "Enter A Number: " number
sum=0

for ((i = 1; i <= $(($number / 2 + 1)); i++))
do
    if (($number % $i == 0))
    then
        sum=$(($sum + i))
    fi
done
if (($sum == $number))
then
    echo "Perfect Number"
else
    echo "Not A Perfect Number"
fi
```

Loop/Prime.sh

Procgram To Check Whether A Number Is Prime Or Not

```
# Procgram To Check Whether A Number Is Prime Or Not

read -p "Enter A Number: " number
flag=0
for ((i=2; i < $(($number / 2 + 1)); i++))
do
    if [ $(($number % $i)) -eq 0 ]
    then
        flag=1
    fi
done

if [ $flag -eq 0 ]
then
    echo "Prime Number"
else
    echo "Not A Prime Number"
fi
```

Loop/ReverseNumber.sh

Program To Reverse A Number

```
# Program To Reverse A Number

read -p "Enter A Number: " number
temp=$number
reverse=0

while [ $number -gt 0 ]
do
    digit=$(($number % 10))
    number=$(($number / 10))
    reverse=$(($reverse * 10 + digit))
done

printf "Reverse: %d\n" $reverse
```

Loop/StrongNumber.sh

Program To Check Whether A Number Is Strong Number Or Not

```
# Program To Check Whether A Number Is Strong Number Or Not
# 1! + 4! + 5! = 145

read -p "Enter A Number: " number
backup=$number
sum=0
while (($number >  0))
do
    digit=$(($number % 10))
    number=$(($number / 10))
    fact=1
    for ((i = 1; i <= digit; i++))
    do
        fact=$(($fact * $i))
    done
    sum=$(($sum + $fact))
done

if (( $backup == $sum ))
then
    echo "Strong Number"
else
    echo "Not A Strong Number"
fi
```

Loop/SumOfDigits.sh

Program To Sum Of Digits Of A Number

```
# Program To Sum Of Digits Of A Number

read -p "Enter A Number: " number
sum=0
```

```
5
6   while [ $number -gt 0 ]
7   do
8       digit=$(($number % 10))
9       sum=$(($sum + $digit))
10      number=$(($number / 10))
11  done
12
13  printf "Sum Of Digits: %d\n" $sum
```

Loop/Table.sh

Program To Print Table Of A Number

```
1   # Program To Print Table Of A Number
2
3   read -p "Enter A Number: " number
4   for i in {1..10}
5   do
6       printf "%d X %d = %d\n" $number $i $(($number * $i))
7   done
```

String/Capitalize.sh

Program To Capitalize Each Word Of The String

```
1   # Program To Capitalize Each Word Of The String
2
3   read -p "Enter A String: " str
4   len=${#str}
5   result=""
6   char=""
7
8   for ((i = 0; i < $len; i++))
9   do
10      prevChar=$char
11      char=${str:$i:1}
12      ascii=`printf "%d" "'$char"`
13      if [ "$prevChar" = "" -o $ascii -ge 97 -a $ascii -le 122 -a "$prevChar" = " " ]
14      then
15          ascii=$(($ascii - 32))
16          result=$result`printf \\\\$(printf "%o" $ascii)`
17      else
18          result=$result`printf \\\\$(printf "%o" $ascii)`
19      fi
20  done
21
22  echo "Capitalized: $result"
```

String/Compare.sh

Program To Compare Two Strings

```
1   # Program To Compare Two Strings
2
3   read -p "Enter First String: " str1
```

```
read -p "Enter Second String: " str2

len1=${#str1}
len2=${#str2}

i=0
flag=0

while [ $flag -eq 0 ]
do
    if [ $i -lt $len1 ]
    then
        char1=${str1:i:1}
        ascii1=`printf "%d" "'$char1"`
    else
        ascii1=0
        flag=1
    fi

    if [ $i -lt $len2 ]
    then
        char2=${str2:i:1}
        ascii2=`printf "%d" "'$char2"`
    else
        ascii2=0
        flag=1
    fi

    diff=$(($ascii2 - $ascii1))

    if (( $diff != 0 || $flag == 1))
    then
        echo $diff
        flag=1
    fi
    i=$((i + 1))
done
```

String/Count.sh

Program To Count Number Of Alphabets, Numbers, Whitespace & Special Character In A String

```
# Program To Count Number Of Alphabets, Numbers, Whitespace & Special Character In A String

read -p "Enter A String: " str
len=${#str}
alpha=0
num=0
special=0
whitespace=0

for ((i = 0; i < $len; i++))
do
    char=${str:$i:1}
    if [[ $char =~ [[:alpha:]] ]]
        then alpha=$(($alpha  + 1))
    elif [[ $char =~ [[:digit:]] ]]
        then num=$(($num + 1))
    elif [[ $char =~ [[:space:]] ]]
        then whitespace=$(($whitespace + 1))
    else
        special=$(($special + 1))
    fi
done

echo "Alphabets: $alpha"
```

```
25  echo "Numbers: $num"
26  echo "Whitespaces: $whitespace"
27  echo "Special Characters: $special"
```

String/Frequency2.sh

Program To Find The Frequency Of Each Alphabet In A String

```
1   # Program To Find The Frequency Of Each Alphabet In  A String
2
3   read -p "Enter A String: " str
4   str=${str^^}
5   alpha=($(printf "0%.0s " {1..26}))
6
7
8   for ((i = 0; i < ${#str}; i++))
9   do
10      char=${str:$i:1}
11      if [[ $char =~ [[:alpha:]] ]]
12      then
13          ascii=`printf "%d" "'$char"`
14          ascii=$(($ascii - 65))
15          alpha[$ascii]=$((${alpha[$ascii]} + 1))
16      fi
17  done
18
19  for ((i = 0; i < 26; i++))
20  do
21      if ((${alpha[$i]} > 0))
22      then
23          char=`printf \\\\$(printf "%o" $(($i + 65)))`
24          printf "%s\t%d\n" $char ${alpha[$i]}
25      fi
26  done
```

String/Frequency.sh

Program To Find Frequency Of A String Or A Character In A Given String

```
1   # Program To Find Frequency Of A String Or A Character In A Given String
2
3   read -p "Enter A String: " str
4   read -p "Enter String To Be Searched: " search
5   searchLen=${#search}
6   max=$((${#str} - $searchLen))
7   count=0
8
9   for ((i = 0; i < max; i++))
10  do
11      if [ "${str:$i:$searchLen}" = "$search" ]
12      then
13          count=$(($count + 1))
14      fi
15  done
16
17  echo "Frequency Of Searched String Is: $count"
```

String/LowerToUpper2.sh

Program To Convert Uppercase String To Lower Case String

```
# Program To Convert Uppercase String To Lower Case String

read -p "Enter A String: " str
echo "Upper Case: ${str^^}"
```

String/LowerToUpper.sh

Program To Convert Uppercase String To Lower Case String

```
# Program To Convert Uppercase String To Lower Case String

read -p "Enter A String: " str
result=""
len=${#str}

for ((i = 0; i < len; i++))
do
    char=${str:i:1}
    ascii=`printf "%d" "'$char"`
    if [ $ascii -ge 97 -a $ascii -le 122 ]
    then
        ascii=$(($ascii - 32))
        result=$result`printf \\\\$(printf "%o" $ascii)`

    else
        result=$result`printf \\\\$(printf "%o" $ascii)`
        # $(printf "%o" $ascii) Converts $ascii To Octal
        # printf \\octal Converts An Octal Number To Character
    fi
done

echo "Upper Case: $result"
```

String/MaxWord.sh

Programs To Input A String And Find The Largest And Smallest Word

```
# Programs To Input A String And Find The Largest And Smallest Word

read -p "Enter A String: " str
str=$str" "
maxWord=""
maxLen=-1
minWord=""
minLen=-1
currentWord=""

for ((i = 0; i < ${#str}; i++))
do
    currentWord=$currentWord${str:$i:1}
    if [[ ${str:$i:2}  =~ [^[:space:]][[:space:]] ]]
    then
        currentWordLen=${#currentWord}
        if [ $currentWordLen -lt $minLen -o $minLen -eq -1 ]
        then
             minWord=$currentWord
             minLen=$currentWordLen
```

```
        fi

        if [ $currentWordLen -gt $maxLen -o $maxLen -eq -1 ]
        then
            maxWord=$currentWord
            maxLen=$currentWordLen
        fi
        currentWord=""
    fi
done

echo "Largest Word: $maxWord"
echo "Smallest Word: $minWord"
```

String/PalindromeString.sh

Program To Check Whether The String Is Palindrome

```
# Program To Check Whether The String Is Palindrome

read -p "Enter A String: " string

len=${#string}
reverse=""

for ((i = $len - 1; i >= 0; i--))
do
    reverse="$reverse${string:$i:1}"
done

if [ "$string" = "$reverse" ]
then
    echo "String Is Palindrome"
else
    echo "String Is Not Palindrome"
fi
```

String/RepeatedString.sh

Command Line Program To Generate A Repeated String N Times With A Given Delimiter

```
# Command Line Program To Generate A Repeated String N Times With A Given Delimiter
str=$1
times=$2
delimiter=$3

if [ "$2" = "" ]; then times=0; fi

for ((i = 0; i < $times; i++))
do
    printf "%s%s" "$str" "$delimiter"
done

echo
```

String/Replace.sh

Program To Replace All Occurances Of A String Or Character By

```
# Program To Replace All Occurances Of A String Or Character By
# Another String Or Character In A Given String

read -p "Enter A String: " str
read -p "Enter String To Be Searched: " search
read -p "Enter String To Be Replaced With: " replace

searchLen=${#search}
max=${#str}
result=""
for ((i = 0; i < max; i++))
do
    slice=${str:$i:$searchLen}
    if [ "$slice" = "$search" ]
    then
        result=$result$replace
        i=$(($i + $searchLen - 1))
    else
        result=$result${str:$i:1}
    fi
done

echo "Result: $result"
```

String/Slice.sh

Program To Slice A String

```
# Program To Slice A String

read -p "Enter A String: " str
read -p "Enter Starting Index: " i
read -p "Enter Final Index: " j

len=${#str}
slice=""

if (( $i < 0 || $i >= len || $j < 0 || $j >= len || $j < $i ))
then
    echo "Invalid Index"
else
    for ((k = i; k <= j; k++))
    do
        slice=$slice${str:k:1}
    done
fi


echo "Slice: $slice"
```

String/StringLength.sh

Program To Find Out Length Of A String

```
# Program To Find Out Length Of A String

read -p "Enter A String: " str
```

```
len=0

while [ "${str:len:1}" != "" ]
do
    len=$(($len + 1))
done

echo "Length: $len"
```

String/StringReverse.sh

Program To Reverse A String

```
# Program To Reverse A String

read -p "Enter A String: " string

len=${#string}
reverse=""
for ((i = $len - 1; i >= 0; i--))
do
    reverse="$reverse${string:$i:1}"
done

printf "Reverse: %s\n" "$reverse"

```

String/StringSearch.sh

Program To Search A String In A Given String

```
# Program To Search A String In A Given String

read -p "Enter A String: " str
read -p "Enter String To Be Searched: " search
searchLen=${#search}
max=$((${#str} - $searchLen))
index=-1

for ((i = 0; i < max; i++))
do
    if [ "${str:$i:$searchLen}" = "$search" ]
    then
        index=$i
        break
    fi
done

if (( $index != -1 ))
then
    echo "Searched String Is Present At Index: $index"
else
    echo "Not Found"
fi
```

String/ToggleCase.sh

Program To Toggle Case Of Every Character

```
# Program To Toggle Case Of Every Character

read -p "Enter A String: " str
result=""
len=${#str}

for ((i = 0; i < len; i++))
do
    char=${str:i:1}
    ascii=`printf "%d" "'$char"`
    if [ $ascii -ge 97 -a $ascii -le 122 ]
    then
        ascii=$(($ascii - 32))
        result=$result`printf \\\\$(printf "%o" $ascii)`

    elif [ $ascii -ge 65 -a $ascii -le 90 ]
    then
        ascii=$(($ascii + 32))
        result=$result`printf \\\\$(printf "%o" $ascii)`
    else
        result=$result`printf \\\\$(printf "%o" $ascii)`
    fi
done

echo "Toggle Case: $result"
```

String/Trim.sh

Program To Trim Extra Whitespaces In A String

```
# Program To Trim Extra Whitespaces In A String

read -p "Enter A String: " str
result=""

for ((i = 0; i < ${#str}; i++))
do
    if [[ ${str:$i:2}  =~ [[:space:]][[:space:]] ]]
    then
        continue
    else
        result=$result${str:$i:1}
    fi
done

echo "Result: $result"
```

String/UpperToLower2.sh

Program To Convert Uppercase String To Lower Case String

```
# Program To Convert Uppercase String To Lower Case String

read -p "Enter A String: " str
echo "Lower Case: ${str,,}"
```

String/UpperToLower.sh

Program To Convert Uppercase String To Lower Case String

```
# Program To Convert Uppercase String To Lower Case String

read -p "Enter A String: " str
result=""
len=${#str}

for ((i = 0; i < len; i++))
do
    char=${str:i:1}
    ascii=`printf "%d" "'$char"`
    if [ $ascii -ge 97 -a $ascii -le 122 ]
    then
        ascii=$(($ascii - 32))
        result=$result`printf \\\\$(printf "%o" $ascii)`

    else
        result=$result`printf \\\\$(printf "%o" $ascii)`
        # $(printf "%o" $ascii) Converts $ascii To Octal
        # printf \\octal Converts An Octal Number To Character
    fi
done

echo "Upper Case: $result"
```

String/Vowel.sh

Program To Count Total Number Of Alphabets and Consonants In A String

```
# Program To Count Total Number Of Alphabets and Consonants In A String

read -p "Enter A String: " str
vowels=0
consonants=0

for (( i = 0; i < ${#str}; i++))
do
    alpha=${str:$i:1}
    if [[ $alpha =~ [aeiouAEIOU] ]]
    then
        vowels=$(($vowels + 1))
    elif [[ $alpha =~ [bcdfghjklmnpqrstvwxyzBCDFGHJKLMNPQRSTVWXYZ] ]]
    then
        consonants=$(($consonants + 1))
    fi
done


echo "Vowels: $vowels"
echo "Consonants: $consonants"
```

String/Words.sh

Program To Input A Sentence And Count The Number Of Words

```
# Program To Input A Sentence And Count The Number Of Words

read -p "Enter A Sentence: " str
len=${#str}
words=0

if [ "$str" != "" ]
then
    for ((i = 0; i < $len - 1; i++))
    do
        if [[ ${str:$i:2}  =~ [[:space:]][^[:space:]] ]]
        then
            words=$(($words + 1))
        fi
    done
else
    words=-1
fi
echo "Words: $(($words + 1))"
```