$ Shell Scripting

Submitted By

Bharat Singh Rajput

BCA VI

717042

If Else

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source EvenOdd.sh
Enter A Number: 6
Even Number
chirag@chirag-desktop:~/Programs/IfElse$ source EvenOdd.sh
Enter A Number: 9
Odd Number
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source Greater3.sh
Enter A : 78
Enter B : 32
Enter C : 10
```

```
A Is Greatest
chirag@chirag-desktop:~/Programs/IfElse$ source Greater3.sh
Enter A : 67
Enter B : 55
Enter C : 101
C Is Greatest
chirag@chirag-desktop:~/Programs/IfElse$ source Greater3.sh
Enter A : 56
Enter B : 89
Enter C : 33
B Is Greatest
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source LeapYear.sh
Enter Year: 1920
Leap Year
chirag@chirag-desktop:~/Programs/IfElse$ source LeapYear.sh
Enter Year: 2017
Not A Leap  Year
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source LUcase.sh
Enter An Alphabet: F
Uppercase Alphabet
```

```
chirag@chirag-desktop:~/Programs/IfElse$ source LUcase.sh
Enter An Alphabet: g
Lowercase Alphabet
chirag@chirag-desktop:~/Programs/IfElse$ source LUcase.sh
Enter An Alphabet: 5
Not An Alphabet
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source PositiveNegative.sh
Enter A Number: 78
Positive Number
chirag@chirag-desktop:~/Programs/IfElse$ source PositiveNegative.sh
Enter A Number: -101
Negative Number
chirag@chirag-desktop:~/Programs/IfElse$ source PositiveNegative.sh
Enter A Number: 0
Zero
```

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

```
# Program To Determine In 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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source Quadrant.sh
Enter X Co-ordinate: 3
Enter Y Co-ordinate: 7
First Quadrant
chirag@chirag-desktop:~/Programs/IfElse$ source Quadrant.sh
Enter X Co-ordinate: -8
Enter Y Co-ordinate: -1
Third Quadrant
chirag@chirag-desktop:~/Programs/IfElse$ source Quadrant.sh
Enter X Co-ordinate: -2
Enter Y Co-ordinate: 8
Second Quadrant
chirag@chirag-desktop:~/Programs/IfElse$ source Quadrant.sh
Enter X Co-ordinate: 8
Enter Y Co-ordinate: -1
Fourth Quadrant
chirag@chirag-desktop:~/Programs/IfElse$ source Quadrant.sh
Enter X Co-ordinate: 0
Enter Y Co-ordinate: 2
Point Lies On Axes
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source Roots.sh
Enter The Value Of a: 6
Enter The Value Of b: 4
Enter The Value Of c: 2
Discriminant: -32
Roots Are Imaginary
chirag@chirag-desktop:~/Programs/IfElse$ source Roots.sh
Enter The Value Of a: 8
Enter The Value Of b: 31
Enter The Value Of c: 7
```

```
Discriminant: 737
Roots Are Real And Distinct
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source Triangle.sh
Enter side A : 6
Enter side B : 8
Enter side C : 4
Scalene Triangle
chirag@chirag-desktop:~/Programs/IfElse$ source Triangle.sh
Enter side A : 8
Enter side B : 8
Enter side C : 8
Equilateral Triangle
chirag@chirag-desktop:~/Programs/IfElse$ source Triangle.sh
Enter side A : 60
Enter side B : 30
Enter side C : 100
Invalid Triangle
chirag@chirag-desktop:~/Programs/IfElse$ source Triangle.sh
Enter side A : 8
Enter side B : 8
Enter side C : 15
Isoceles Triangle
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source Vowel.sh
Enter An Alphabet: a
Vowel
chirag@chirag-desktop:~/Programs/IfElse$ source Vowel.sh
Enter An Alphabet: t
Consonant
chirag@chirag-desktop:~/Programs/IfElse$ source Vowel.sh
Enter An Alphabet: 2
Not An Alphabet
```

Program To Input Marks In 5 Subjects And Calculate Grades

```
# Program to input marks of five subjects Physics, Chemistry, Biology, Mathematics
# 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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source Grade.sh
Enter Marks In Physics: 100
Enter Marks In Chemistry: 100
Enter Marks In Biology: 100
Enter Marks In Math: 90
Enter Marks In Computer: 90
Percentage: 96%
Grade A
chirag@chirag-desktop:~/Programs/IfElse$ source Grade.sh
Enter Marks In Physics: 90
Enter Marks In Chemistry: 80
Enter Marks In Biology: 75
Enter Marks In Math: 89
Enter Marks In Computer: 81
Percentage: 83%
Grade B
chirag@chirag-desktop:~/Programs/IfElse$ source Grade.sh
Enter Marks In Physics: 76
Enter Marks In Chemistry: 40
Enter Marks In Biology: 39
Enter Marks In Math: 50
Enter Marks In Computer: 55
Percentage: 52%
Grade E
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/IfElse$ source ProfitLoss.sh
Enter Cost Price: 760
Enter Selling Price: 800
Profit: Rs.40
Profit Percentage: 5.26300%
chirag@chirag-desktop:~/Programs/IfElse$ source ProfitLoss.sh
Enter Cost Price: 500
Enter Selling Price: 520
Profit: Rs.20
Profit Percentage: 4.00000%
```

Loops

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

if [ $sum -eq $temp ]
then
    echo "Armstrong Number"
else
    echo "Not An Armstrong Number"
fi
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Armstrong.sh
Enter A Number: 153
Armstrong Number
chirag@chirag-desktop:~/Programs/Loop$ source Armstrong.sh
Enter A Number: 476
Not An Armstrong Number
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source DecimalToBinary.sh
Enter A Number: 67
Binary Equivalent: 1000011
chirag@chirag-desktop:~/Programs/Loop$ source DecimalToBinary.sh
Enter A Number: 16
Binary Equivalent: 10000
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Factorial.sh
Enter A Number: 5
Factorial: 120
chirag@chirag-desktop:~/Programs/Loop$ source Factorial.sh
Enter A Number: 10
Factorial: 3628800
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Fibonacci.sh
Enter N: 9
0 1 1 2 3 5 8 13 21
```

Program To Check Whether A Number Palindrome Or Not

```
# Program To Check Whether A Number Palindrome Or Not

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

if [ $reverse -eq $temp ]
then
    echo "Palindrome Number"
else
    echo "Not An Palindrome Number"
fi
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source PalindromeNumber.sh
Enter A Number: 195591
Palindrome Number
chirag@chirag-desktop:~/Programs/Loop$ source PalindromeNumber.sh
Enter A Number: 123
Not An Palindrome Number
```

Program To Print Pattern

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

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Pattern1.sh
Enter The Number Of Rows: 7
*
**
***
****
*****
******
*******
```

Program To Print Pattern

```
# 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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Pattern2.sh
Enter The Number Of Rows: 8
********
*******
******
*****
****
***
```

```
**
*
```

Program To Print Pattern

```
# 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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Pattern3.sh
Enter The Number Of Rows: 10
         *
        ***
       *****
      *******
     *********
    ***********
   *************
  ***************
 *****************
*******************
```

Program To Check Whether A Number Is Prime Or Not

```
# Program 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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Prime.sh
Enter A Number: 87
Not A Prime Number
chirag@chirag-desktop:~/Programs/Loop$ source Prime.sh
Enter A Number: 73
Prime Number
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source ReverseNumber.sh
Enter A Number: 12634
Reverse: 43621
```

Program To Sum Of Digits Of A Number

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

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

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

```
    number=$(($number / 10))
done

printf "Sum Of Digits: %d\n" $sum
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source SumOfDigits.sh
Enter A Number: 12367
Sum Of Digits: 19
```

Program To Print Table Of A Number

```
# Program To Print Table Of A Number

read -p "Enter A Number: " number
for i in {1..10}
do
    printf "%d X %d = %d\n" $number $i $(($number * $i))
done
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Table.sh
Enter A Number: 19
19 X 1 = 19
19 X 2 = 38
19 X 3 = 57
19 X 4 = 76
19 X 5 = 95
19 X 6 = 114
19 X 7 = 133
19 X 8 = 152
19 X 9 = 171
19 X 10 = 190
```

Program To Input N Numbers And Calculate Their Average

```
# Program To Input N Numbers And Calculate Their Average

flag=1
length=0
sum=0

while [ $flag -eq 1 ]
do
    read -p "Enter A Number: " input
    sum=$($sum + $input))
    length=$($length + 1))
    read -n1 -p "Press 1 To Enter More Or Any Other Key To Exit: " flag
    echo
done

average=`echo "scale=4; $sum / $length" | bc`
```

```
echo "Average: $average"
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Average.sh
Enter A Number: 9
Press 1 To Enter More Or Any Other Key To Exit: 1
Enter A Number: 18
Press 1 To Enter More Or Any Other Key To Exit: 1
Enter A Number: 12
Press 1 To Enter More Or Any Other Key To Exit: 0
Average: 13.0000
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source APseries.sh
Enter The Value Of First Term (a): 1
Enter The Value Of Common Difference (d): 1
Enter N: 10
1 2 3 4 5 6 7 8 9 10
chirag@chirag-desktop:~/Programs/Loop$ source APseries.sh
Enter The Value Of First Term (a): 2
Enter The Value Of Common Difference (d): 5
Enter N: 10
2 7 12 17 22 27 32 37 42 47
```

Program To Convert Binary To Decimal

```
# Program To Convert Binary To Decimal

read -p "Enter A Binary Number: " binary

len=${#binary}
j=1
result=0

for ((i = len - 1; i >= 0; i--))
do
    digit=${binary:i:1}
```

```
    result=$(($result + $j * $digit))
    j=$(($j * 2))
done

echo "Decimal: $result"
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source BinaryToDecimal.sh
Enter A Binary Number: 101010
Decimal: 42
chirag@chirag-desktop:~/Programs/Loop$ source BinaryToDecimal.sh
Enter A Binary Number: 111
Decimal: 7
chirag@chirag-desktop:~/Programs/Loop$ source BinaryToDecimal.sh
Enter A Binary Number: 100
Decimal: 4
chirag@chirag-desktop:~/Programs/Loop$ source BinaryToDecimal.sh
Enter A Binary Number: 1111
Decimal: 15
```

Program To Convert Binary To Hexadecimal

```
# Program To Convert Binary To Hexadecimal

read -p "Enter A Binary Number: " binary
len=${#binary}
j=1
temp=0
result=""

for ((i = len - 1; i >= 0; i--))
do
    digit=${binary:i:1}
    temp=$(($temp + $digit * $j))
    j=$(($j * 2))
    if (($j == 16 || i == 0))
    then
        if (($temp < 10)); then result="$temp$result";
        elif (($temp == 10)); then result="A$result"
        elif (($temp == 11)); then result="B$result"
        elif (($temp == 12)); then result="C$result"
        elif (($temp == 13)); then result="D$result"
        elif (($temp == 14)); then result="E$result"
        elif (($temp == 15)); then result="F$result"; fi
        j=1
        temp=0
    fi
done

echo "Hexadecimal: $result"
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source BinaryToHex.sh
Enter A Binary Number: 1010101100001011
Hexadecimal: AB0B
chirag@chirag-desktop:~/Programs/Loop$ source BinaryToHex.sh
```

```
Enter A Binary Number: 110011110001010100111
Hexadecimal: 19E2A7
```

Program To Convert Binary To Octal

```
# Program To Convert Binary To Octal

read -p "Enter A Binary Number: " binary
len=${#binary}
j=1
temp=0
result=""

for ((i = len - 1; i >= 0; i--))
do
    digit=${binary:i:1}
    temp=$(($temp + $digit * $j))
    j=$(($j * 2))
    if (($j == 8 || i == 0))
    then
        result="$temp$result";
        j=1
        temp=0
    fi
done

echo "Octal: $result"
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source BinaryToOctal.sh
Enter A Binary Number: 111111
Octal: 77
chirag@chirag-desktop:~/Programs/Loop$ source BinaryToOctal.sh
Enter A Binary Number: 1101011100101
Octal: 15345
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source DecimalToHex.sh
Enter A Number: 15
Hexadecimal Equivalent: F
chirag@chirag-desktop:~/Programs/Loop$ source DecimalToHex.sh
Enter A Number: 321
Hexadecimal Equivalent: 141
chirag@chirag-desktop:~/Programs/Loop$ source DecimalToHex.sh
Enter A Number: 674
Hexadecimal Equivalent: 2A2
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source DecimalToOctal.sh
Enter A Number: 10
Octal Equivalent: 12
chirag@chirag-desktop:~/Programs/Loop$ source DecimalToOctal.sh
Enter A Number: 1235657
Octal Equivalent: 4555311
chirag@chirag-desktop:~/Programs/Loop$ source DecimalToOctal.sh
Enter A Number: 9999
Octal Equivalent: 23417
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source StrongNumber.sh
Enter A Number: 145
Strong Number
chirag@chirag-desktop:~/Programs/Loop$ source StrongNumber.sh
Enter A Number: 146
Not A Strong Number
chirag@chirag-desktop:~/Programs/Loop$ source StrongNumber.sh
Enter A Number: 40585
Strong Number
```

Program To Find Factors Of A Number

```
# Program To Find Factors Of A Number

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
```

Output

```
Enter A Number: 431
Factors: 1 431
chirag@chirag-desktop:~/Programs/Loop$ source Factors.sh
Enter A Number: 83
Factors: 1 83
chirag@chirag-desktop:~/Programs/Loop$ source Factors.sh
```

```
Enter A Number: 84
Factors: 1 2 3 4 6 7 12 14 21 28 42 84
chirag@chirag-desktop:~/Programs/Loop$ source Factors.sh
Enter A Number: 87
Factors: 1 3 29 87
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Perfect.sh
Enter A Number: 23
Not A Perfect Number
chirag@chirag-desktop:~/Programs/Loop$ source Perfect.sh
Enter A Number: 28
Perfect Number
chirag@chirag-desktop:~/Programs/Loop$ source Perfect.sh
Enter A Number: 495
Not A Perfect Number
chirag@chirag-desktop:~/Programs/Loop$ source Perfect.sh
Enter A Number: 496
```

Program To Calculate Nth Power Of A Number

```
# Program To Calculate Nth Power Of A Number

read -p "Enter A Number: " number
read -p "Enter N: " n

result=1

for ((i = 0; i < $n; i++))
do
    result=$(($result * $number))
done
```

```
echo "Result: $result"
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source NthPower.sh
Enter A Number: 3
Enter N: 2
Result: 9
chirag@chirag-desktop:~/Programs/Loop$ source NthPower.sh
Enter A Number: 2
Enter N: 5
Result: 32
```

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"
```

Output

```
Enter First Number: 8
Enter Second Number: 10
GCD: 2
chirag@chirag-desktop:~/Programs/Loop$ source EuclideanGCD.sh
Enter First Number: 35
Enter Second Number: 10
GCD: 5
chirag@chirag-desktop:~/Programs/Loop$ source EuclideanGCD.sh
Enter First Number: 50
Enter Second Number: 100
GCD: 50
```

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

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

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source LCM.sh
Enter First Number: 6
Enter Second Number: 10
LCM: 30
chirag@chirag-desktop:~/Programs/Loop$ source LCM.sh
Enter First Number: 6
Enter Second Number: 7
LCM: 42
chirag@chirag-desktop:~/Programs/Loop$ source LCM.sh
Enter First Number: 12
Enter Second Number: 30
LCM: 60
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source GPseries.sh
Enter The Value Of First Term (a): 2
Enter The Value Of Common Ratio (r): 5
Enter N: 4
2 10 50 250
chirag@chirag-desktop:~/Programs/Loop$ source GPseries.sh
Enter The Value Of First Term (a): 2
Enter The Value Of Common Ratio (r): 3
Enter N: 5
2 6 18 54 162
chirag@chirag-desktop:~/Programs/Loop$ source GPseries.sh
Enter The Value Of First Term (a): 2
Enter The Value Of Common Ratio (r): 2
```

```
Enter N: 4
2 4 8 16
```

Program To Print Floyd Triangle Upto N Rows

```bash
# 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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source FloydTriangle.sh
Enter The Number Of Rows: 7
1
01
101
0101
10101
010101
1010101
```

Program To Print The Sum Of Factorial Series Upto N Terms

```bash
# 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"
```

Output

```
g@chirag-desktop:~/Programs/Loop$ source FactorialSeries.sh
Enter N: 3
Sum: 9
chirag@chirag-desktop:~/Programs/Loop$ source FactorialSeries.sh
Enter N: 4
Sum: 33
```

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

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

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
```

Output

```
chirag@chirag-desktop:~/Programs/Loop$ source Circle.sh
Enter Radius: 7
Circumference: 43.982260
Area: 153.937910
chirag@chirag-desktop:~/Programs/Loop$ source Circle.sh
Enter Radius: 11
Circumference: 69.114980
Area: 380.132390
```

Strings

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
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source PalindromeString.sh
Enter A String: madam
String Is Palindrome
chirag@chirag-desktop:~/Programs/String$ source PalindromeString.sh
Enter A String: chirag
String Is Not Palindrome
```

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"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source StringReverse.sh
Enter A String: chirag
Reverse: garihc
chirag@chirag-desktop:~/Programs/String$ source StringReverse.sh
Enter A String: Hello World!
Reverse: !dlroW olleH
```

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))"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source Words.sh
Enter A Sentence: Hello World!
Words: 2
chirag@chirag-desktop:~/Programs/String$ source Words.sh
Enter A Sentence: My College Is National P G College
Words: 7
```

Program To Convert A Lowercase String To Upper Case

```
# Program To Convert Lowercase String To UpperCase 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"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source LowerToUpper.sh
Enter A String: National PG College
Upper Case: NATIONAL PG COLLEGE
chirag@chirag-desktop:~/Programs/String$ source LowerToUpper.sh
Enter A String: Chirag Singh Rajput
Upper Case: CHIRAG SINGH RAJPUT
```

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 65 -a $ascii -le 90 ]
    then
        ascii=$(($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 "Lower Case: $result"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source UpperToLower.sh
Enter A String: CHIRAG SINGH RAJPUT
Lower Case: chirag singh rajput
chirag@chirag-desktop:~/Programs/String$ source UpperToLower.sh
Enter A String: MY PHONE IS 1234
Lower Case: my phone is 1234
```

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"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source ToggleCase.sh
Enter A String: Chirag Singh Rajput
Toggle Case: cHIRAG sINGH rAJPUT
chirag@chirag-desktop:~/Programs/String$ source ToggleCase.sh
Enter A String: Nation PG cOllEge
Toggle Case: nATION pg CoLLeGE
```

Program To Convert LowerCase String To UpperCase String

```
# Program To Convert LowerCase String To UpperCase String
# Method-2 Using Echo

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

Output

```
chirag@chirag-desktop:~/Programs/String$ source LowerToUpper2.sh
Enter A String: Chirag
Upper Case: CHIRAG
chirag@chirag-desktop:~/Programs/String$ source LowerToUpper2.sh
Enter A String: National PG College
Upper Case: NATIONAL PG COLLEGE
```

Program To Convert Uppercase String To Lower Case String

```
# Program To Convert Uppercase String To Lower Case String
# Method-2 Using Echo

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

Output

```
chirag@chirag-desktop:~/Programs/String$ source UpperToLower2.sh
Enter A String: CHIRAG SINGH RAJPUT
Lower Case: chirag singh rajput
chirag@chirag-desktop:~/Programs/String$ source UpperToLower2.sh
Enter A String: National PG College
Lower Case: national pg colleg
```

Program To Calculate 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"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source StringLength.sh
Enter A String: Chirag Singh Rajput
Length: 19
```

```
chirag@chirag-desktop:~/Programs/String$ source StringLength.sh
Enter A String: NATIONAL P G COLLEGE
Length: 20
```

Program To Capitalize Each Word Of The String

```
# Program To Capitalize Each Word Of The String

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

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

echo "Capitalized: $result"
```

Output

```
Capitalized: Nation PG College
chirag@chirag-desktop:~/Programs/String$ source Capitalize.sh
Enter A String: chirag singh rajput
Capitalized: Chirag Singh Rajput
chirag@chirag-desktop:~/Programs/String$ source Capitalize.sh
Enter A String: my college is national p g college
Capitalized: My College Is National P G College
```

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"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source Slice.sh
Enter A String: Chirag Singh Rajput
Enter Starting Index: 4
Enter Final Index: 9
Slice: ag Sin
chirag@chirag-desktop:~/Programs/String$ source Slice.sh
Enter A String: World
Enter Starting Index: 3
Enter Final Index: 3
Slice: l
```

Program To Compare Two Strings

```
# Program To Compare Two Strings

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
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source Compare.sh
Enter First String: apple
Enter Second String: apples
115
chirag@chirag-desktop:~/Programs/String$ source Compare.sh
Enter First String: chirag
Enter Second String: Chirag
-32
chirag@chirag-desktop:~/Programs/String$ source Compare.sh
Enter First String: abc
Enter Second String: abc
0
```

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

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

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

Output

```
chirag@chirag-desktop:~/Programs/String$ source Count.sh
Enter A String: My Phone Number Is 9839523456 and my House Number is 1/823/254-sec
Alphabets: 42
Numbers: 17
Whitespaces: 11
Special Characters: 3
```

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"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source Vowel.sh
Enter A String: Chirag Singh Rajput
Vowels: 5
Consonants: 12
chirag@chirag-desktop:~/Programs/String$ source Vowel.sh
Enter A String: National PG College
Vowels: 7
Consonants: 10
```

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
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source StringSearch.sh
Enter A String: National PG College
Enter String To Be Searched: Col
Searched String Is Present At Index: 12
chirag@chirag-desktop:~/Programs/String$ source StringSearch.sh
Enter A String: Chirag Singh Rajput
Enter String To Be Searched: Sing
Searched String Is Present At Index: 7
```

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

read -p "Enter A String: " str
read -p "Enter String To Be Searched: " search
searchLen=${#search}
max=$((${#str} - $searchLen))
count=0

for ((i = 0; i < max; i++))
do
    if [ "${str:$i:$searchLen}" = "$search" ]
    then
        count=$(($count + 1))
    fi
done

echo "Frequency Of Searched String Is: $count"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source Frequency.sh
Enter A String: Chirag Singh Rajput
Enter String To Be Searched: a
Frequency Of Searched String Is: 2
chirag@chirag-desktop:~/Programs/String$ source Frequency.sh
Enter A String: The Taj Mahal Is One Of The Seven Wonders Of The World
Enter String To Be Searched: The
Frequency Of Searched String Is: 3
```

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

```
# 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"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source Replace.sh
Enter A String: Cat Is My Favourite Animal Because Cats Are Very Cute.
Enter String To Be Searched: Cat
Enter String To Be Replaced With: Baboon
Result: Baboon Is My Favourite Animal Because Baboons Are Very Cute.
chirag@chirag-desktop:~/Programs/String$ source Replace.sh
Enter A String: Chirag Singh Rajput
Enter String To Be Searched: u
Enter String To Be Replaced With: oo
Result: Chirag Singh Rajpoot
```

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"
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source MaxWord.sh
Enter A String: Hello My Name Is Chirag Singh Rajpoot
Largest Word:  Rajpoot
Smallest Word:  My
chirag@chirag-desktop:~/Programs/String$ source MaxWord.sh
Enter A String: National PG College
Largest Word: National
Smallest Word:  PG
```

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

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"

```
#### Output
```bash
chirag@chirag-desktop:~/Programs/String$ source Trim.sh
Enter A String:       Chirag         Singh      Rajput
Result: Chirag Singh Rajput
```

Program To Find The Frequency Of Each Alphabet In A String

```
# Program To Find The Frequency Of Each Alphabet In  A String

read -p "Enter A String: " str
str=${str^^}
alpha=($(printf "0%.0s " {1..26}))


for ((i = 0; i < ${#str}; i++))
do
    char=${str:$i:1}
    if [[ $char =~ [[:alpha:]] ]]
    then
        ascii=`printf "%d" "'$char"`
        ascii=$(($ascii - 65))
        alpha[$ascii]=$((${alpha[$ascii]} + 1))
    fi
done

for ((i = 0; i < 26; i++))
do
```
```

```
    if ((${alpha[$i]} > 0))
    then
        char=`printf \\\\$(printf "%o" $(($i + 65)))`
        printf "%s\t%d\n" $char ${alpha[$i]}
    fi
done
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source Frequency2.sh
Enter A String: National PG College
A       2
C       1
E       2
G       2
I       1
L       3
N       2
O       2
P       1
T       1
chirag@chirag-desktop:~/Programs/String$ source Frequency2.sh
Enter A String: Chirag Singh Rajput
A       2
C       1
G       2
H       2
I       2
J       1
N       1
P       1
R       2
S       1
T       1
U       1
```

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 Delimite
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
```

Output

```
chirag@chirag-desktop:~/Programs/String$ source RepeatedString.sh Hello 5 " "
Hello Hello Hello Hello Hello
chirag@chirag-desktop:~/Programs/String$ source RepeatedString.sh 0 10 ,
0,0,0,0,0,0,0,0,0,0,
```

Function To Compute Factorial Of Number Using Loop

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

factorial() {
    number=$1
    factorial=1

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

    echo $factorial
}
```

Output

```
chirag@chirag-desktop:~/Programs/Function$ source FactorialLoop.sh
chirag@chirag-desktop:~/Programs/Function$ factorial 12
479001600
chirag@chirag-desktop:~/Programs/Function$ factorial 0
1
```

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
}
```

Output

```
chirag@chirag-desktop:~/Programs/Function$ source FactorialRecur.sh
chirag@chirag-desktop:~/Programs/Function$ factorial 7
5040
chirag@chirag-desktop:~/Programs/Function$ factorial 0
1
```

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
}
```

Output

```
chirag@chirag-desktop:~/Programs/Function$ source Fibonacci.sh
chirag@chirag-desktop:~/Programs/Function$ fibonacci 7
13
chirag@chirag-desktop:~/Programs/Function$ fibonacci 18
2584
```

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
}
```

Output

```
chirag@chirag-desktop:~/Programs/Function$ source NthPower.sh
chirag@chirag-desktop:~/Programs/Function$ power 3 2
9
chirag@chirag-desktop:~/Programs/Function$ power 9 2
81
```

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
}
```

Output

```
chirag@chirag-desktop:~/Programs/Function$ source EuclideanGCD.sh
chirag@chirag-desktop:~/Programs/Function$ gcd 6 10
2
chirag@chirag-desktop:~/Programs/Function$ gcd 6 9
3
chirag@chirag-desktop:~/Programs/Function$ gcd 100 80
20
```

Program To Compute Permutaions (nPr)

```
# 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))) ))
}
```

Output

```
chirag@chirag-desktop:~/Programs/Function$ source Permutation.sh
chirag@chirag-desktop:~/Programs/Function$ permutation 5 3
60
chirag@chirag-desktop:~/Programs/Function$ permutation 6 2
30
```

Program To Compute Combinations (nCr)

```
# 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) ))
}
```

Output

```
chirag@chirag-desktop:~/Programs/Function$ source Combinations.sh
chirag@chirag-desktop:~/Programs/Function$ combination 6 3
20
chirag@chirag-desktop:~/Programs/Function$ combination 7 2
21
```

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
}
```

Output

```
chirag@chirag-desktop:~/Programs/Function$ source PascalTriangle.sh
chirag@chirag-desktop:~/Programs/Function$ pascal 9
           1
         1  1
        1  2  1
       1  3  3  1
      1  4  6  4  1
     1  5 10 10  5  1
    1  6 15 20 15  6  1
   1  7 21 35 35 21  7  1
  1  8 28 56 70 56 28  8  1
```

Program To Perform Selection Sort

```
# Program To Perform Selection Sort

selectionSort() {
    arr=($*)
    for ((i = 0; i < $(($# - 1)); i++))
    do
        minIndex=$i
        for ((j = $(($i + 1)); j < $#; j++))
        do
```

```
            if ((${arr[$j]} < ${arr[$minIndex]}))
            then
                minIndex=$j
            fi
        done
        temp=${arr[$i]}
        arr[$i]=${arr[$minIndex]}
        arr[$minIndex]=$temp
    done
    echo ${arr[*]}
}
```

Output

```
chirag@chirag-desktop:~/Programs/Array$ source SelectionSort.sh
chirag@chirag-desktop:~/Programs/Array$ selectionSort 100 2 8 44 -9 -88 101 555 0
-88 -9 0 2 8 44 100 101 555
```

Program To Perform Bubble Sort

```
# Program To Perform Bubble Sort

bubbleSort() {
    arr=($*)
    for ((i = 0; i < $(($# - 1)); i++))
    do
        for ((j = 0; j < $(($# - $i - 1)); j++))
        do
            if ((${arr[$j]} > ${arr[$(($j + 1))]}))
            then
                temp=${arr[$j]}
                arr[$j]=${arr[$(($j + 1))]}
                arr[$(($j + 1))]=$temp
            fi
        done

    done
    echo ${arr[*]}
}
```

Output

```
chirag@chirag-desktop:~/Programs/Array$ source BubbleSort.sh
chirag@chirag-desktop:~/Programs/Array$ bubbleSort 100 2 8 44 -9 -88 101 555 0
-88 -9 0 2 8 44 100 101 555
```

Program To Perform Insertion Sort

```
# Program To Perform Insertion Sort

insertionSort() {
    arr=($*)
    for ((i = 1; i < $#; i++))
    do
        key=${arr[$i]}
        j=$(($i - 1))
```

```
        while (($j >= 0 && ${arr[$j]} > $key))
        do
            arr[$(($j + 1))]=${arr[$j]}
            j=$(($j - 1))
        done
        arr[$(($j + 1))]=$key
    done
    echo ${arr[*]}
}
```

Output

```
chirag@chirag-desktop:~/Programs/Array$ source InsertionSort.sh
chirag@chirag-desktop:~/Programs/Array$ insertionSort 100 2 8 44 -9 -88 101 555 0
-88 -9 0 2 8 44 100 101 555
```
