

MALLA REDDY

Institute of Engineering & Technology

(UGC AUTONOMOUS)

(Sponsored by Malla Reddy Educational Society)NBA
Accredited, Affiliated to JNTU, Hyderabad
Maisammaguda, Dhulapally (post via Hakimpet), Sec'Bad-500 014.
Phone: 040-65969674, Cell: 9348161223

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



SCRIPTING LANGUAGES LAB MANUAL(R18)

Subject Code(CS623PE)

LAB MANUAL
MACHINE LEARNING LAB(CS601PC)



Department of Computer Science and Engineering

Document No: MRIET/CSE/LAB MANUAL/SL	Date of Issue Date of Revision	C o m p i l e d b y Dr. NARENDHAR M & G NAGAPPA Verified by	A u t h o r i z e d b y HOD(CSE) & PRINCIPAL
---	---	--	---

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CS623PE: SCRIPTING LANGUAGES LAB (Professional Elective - III)

III Year B.Tech. CSE II-Sem

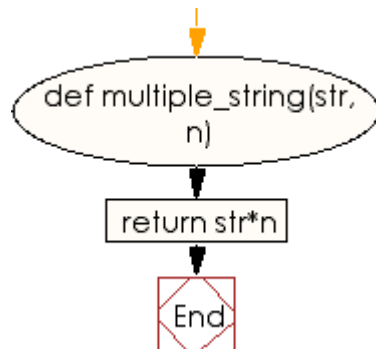
List of Experiments

1. Write a Ruby script to create a new string which is n copies of a given string where n is a non-negative integer
2. Write a Ruby script which accept the radius of a circle from the user and compute the parameter and area.
3. Write a Ruby script which accept the user's first and last name and print them in reverse order with a space between them
4. Write a Ruby script to accept a filename from the user print the extension of that
5. Write a Ruby script to find the greatest of three numbers
6. Write a Ruby script to print odd numbers from 10 to 1
7. Write a Ruby script to check two integers and return true if one of them is 20 otherwise return their sum
8. Write a Ruby script to check two temperatures and return true if one is less than 0 and the other is greater than 100
9. Write a Ruby script to print the elements of a given array
10. Write a Ruby program to retrieve the total marks where subject name and marks of a student stored in a hash
11. Write a TCL script to find the factorial of a number
12. Write a TCL script that multiplies the numbers from 1 to 10
13. Write a TCL script for Sorting a list using a comparison function
14. Write a TCL script to (i) create a list (ii) append elements to the list (iii) Traverse the list (iv) Concatenate the list
15. Write a TCL script to compare the file modified times.
16. Write a TCL script to Copy a file and translate to native format.
17. a) Write a Perl script to find the largest number among three numbers.
b) Write a Perl script to print the multiplication tables from 1-10 using subroutines.
18. Write a Perl program to implement the following list of manipulating functions
a) Shift b) Unshift c) Push
19. a) Write a Perl script to substitute a word, with another word in a string.
b) Write a Perl script to validate IP address and email address.
20. Write a Perl script to print the file in reverse order using command line arguments

1. Write a Ruby script to create a new string which is n copies of a given string where n is a non-negative integer

Aim: Ruby script to create a new string which is n copies of a given string where n is a non-negative integer

Flowchart:



Ruby Code:

```
def multiple_string(str, n)
  return str*n
end
print multiple_string('a', 1),"\\n"
print multiple_string('a', 2),"\\n"
print multiple_string('a', 3),"\\n"
print multiple_string('a', 4),"\\n"
print multiple_string('a', 5),"\\n"
```

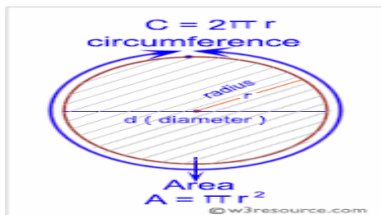
Output:

```
a
aa
aaa
aaaa
aaaaa
```

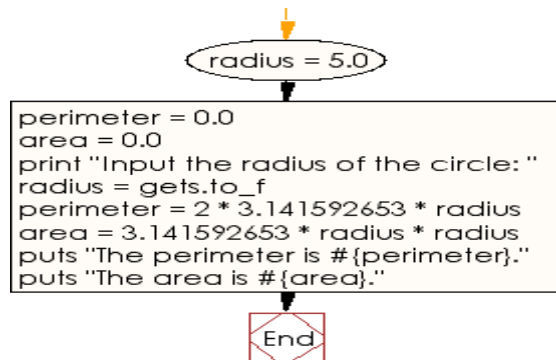
2. Write a Ruby script which accept the radius of a circle from the user and compute the parameter and area.

Aim: Ruby script which accept the radius of a circle from the user and compute the parameter and area.

Block Structure



Flowchart:



Ruby Code:

```
radius = 5.0  
perimeter = 0.0  
area = 0.0  
print "Input the radius of the circle: "  
radius = gets.to_f  
perimeter = 2 * 3.141592653 * radius  
area = 3.141592653 * radius * radius  
puts "The perimeter is #{perimeter}."  
puts "The area is #{area}."
```

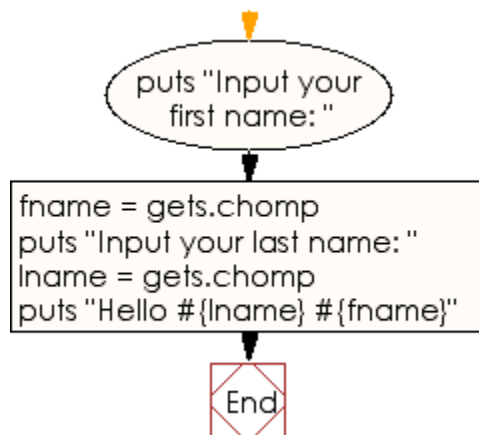
Output:

```
Input the radius of the circle: 2  
The perimeter is 12.566370612.  
The area is 12.566370612.
```

3. Write a Ruby script which accept the user's first and last name and print them in reverse order with a space between them

Aim: Ruby script which accept the user's first and last name and print them in reverse order with a space between them

Flowchart:



Ruby Code:

```
puts "Input your first name: "  
fname = gets.chomp  
puts "Input your last name: "  
lname = gets.chomp  
puts "Hello #{lname} #{fname}"
```

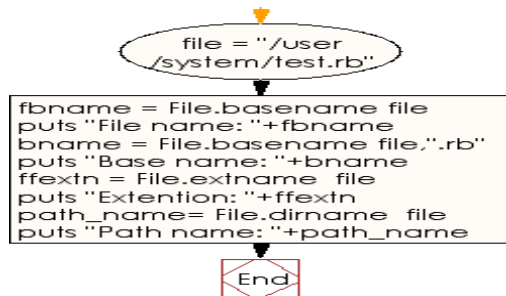
Output:

```
Input your first name:  
Input your last name:  
Hello Naga babu
```

4. Write a Ruby script to accept a filename from the user print the extension of that

Aim: Ruby script to accept a filename from the user print the extension of that

Flowchart:



Ruby Code:

```
file = "/user/system/test.rb"  
# file name  
fbname = File.basename file  
puts "File name: "+fbname  
# basename  
bname = File.basename file, ".rb"  
puts "Base name: "+bname  
# file extention  
ffextn = File.extname file  
puts "Extention: "+ffextn  
# path name  
path_name= File.dirname file  
puts "Path name: "+path_name
```

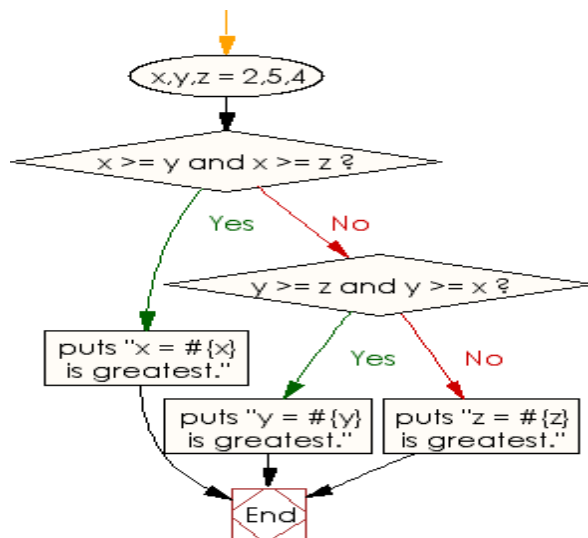
output:

```
File name: test.rb  
Base name: test  
Extention: .rb  
Path name: /user/system
```

5. Write a Ruby script to find the greatest of three numbers

Aim: Ruby script to find the greatest of three numbers

Flowchart:



Ruby Code:

```
x,y,z = 2,5,4
if x >= y and x >= z
  puts "x = #{x} is greatest."
elsif y >= z and y >= x
  puts "y = #{y} is greatest."
else
  puts "z = #{z} is greatest."
end
```

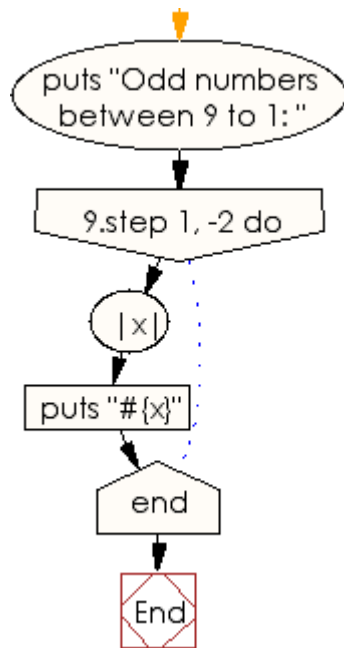
output

y = 5 is greatest.

6. Write a Ruby script to print odd numbers from 10 to 1

Aim: Ruby script to print odd numbers from 10 to 1

Flowchart:



Ruby Code:

```
puts "Odd numbers between 9 to 1: "  
9.step 1, -2 do |x|  
  puts "#{x}"  
end
```

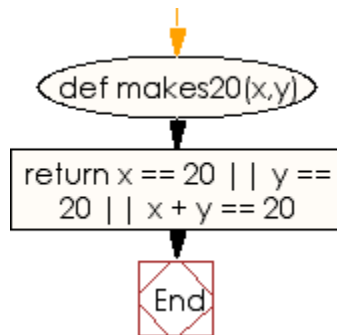
Output

```
Odd numbers between 9 to 1:  
9  
7  
5  
3  
1
```

7. Write a Ruby script to check two integers and return true if one of them is 20 otherwise return their sum

Aim: Ruby script to check two integers and return true if one of them is 20 otherwise return their sum

Flowchart:



Ruby Code:

```
def makes20(x,y)
  return x == 20 || y == 20 || x + y == 20
end
```

```
print makes20(10, 10), "\n"
print makes20(40, 10), "\n"
print makes20(15, 20)
```

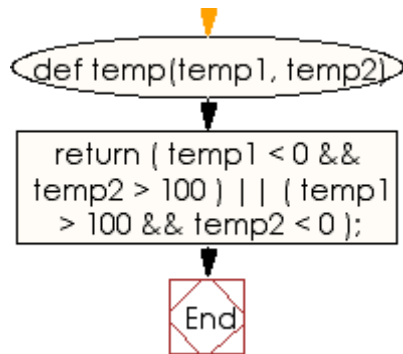
Output:

```
true
false
true
```

8. Write a Ruby script to check two temperatures and return true if one is less than 0 and the other is greater than 100

Aim: Ruby script to check two temperatures and return true if one is less than 0 and the other is greater than 100

Flowchart:



Ruby Code:

```
def temp(temp1, temp2)
  return ( temp1 < 0 && temp2 > 100 ) || ( temp1 > 100 && temp2 < 0 );
end
print temp(110, -1), "\n"
print temp(-1, 110), "\n"
print temp(2, 120)
```

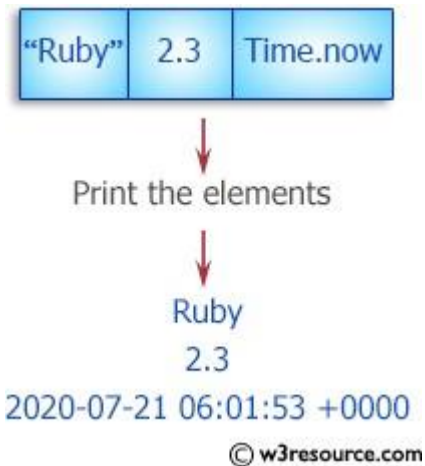
Output

```
true
true
false
```

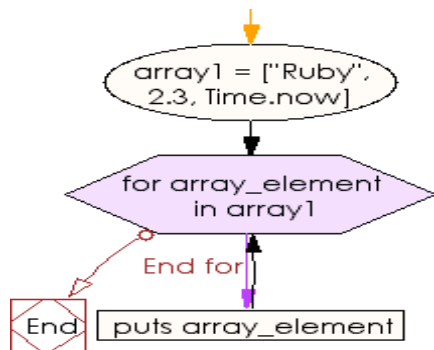
9. Write a Ruby script to print the elements of a given array

Aim: Ruby script to print the elements of a given array

Block Structure



Flowchart:



Ruby Code:

```
array1 = ["Ruby", 2.3, Time.now]
for array_element in array1
  puts array_element
end
```

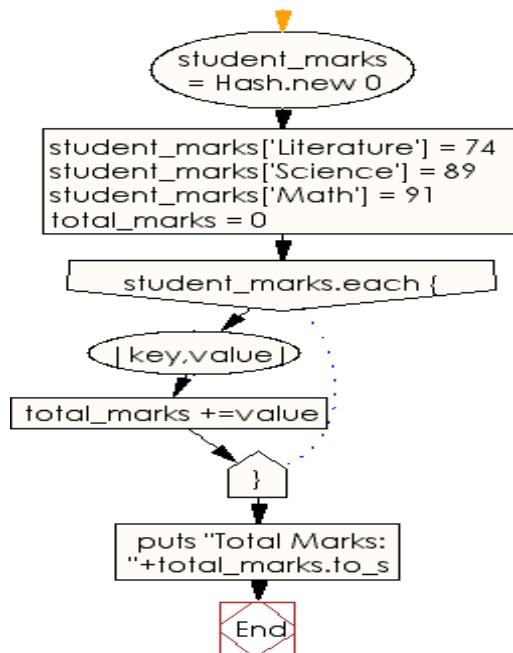
Output

```
Ruby
2.3
2017-12-28 06:01:53 +0000
```

10 Write a Ruby program to retrieve the total marks where subject name and marks of a student stored in a hash

Aim: Ruby program to retrieve the total marks where subject name and marks of a student stored in a hash

Flowchart



Ruby Code:

```
student_marks = Hash.new 0
student_marks['Literature'] = 50
student_marks['Science'] = 40
student_marks['Math'] = 90
total_marks = 0
student_marks.each { |key,value|
  total_marks += value
}
puts "Total Marks: "+total_marks.to_s
```

Output

Total Marks: 180

11) Write a TCL script to find the factorial of a number

Aim: TCL script to find the factorial of a number

Source Code:

```
set fact 1
for {set i 0} {$i <= 10} {incr i} {
    puts "$i! = $fact"
    set fact [expr {$fact * ($i + 1)}]
}
```

Output:

0! = 1

1! = 1

2! = 2

3! = 6

4! = 24

5! = 120

6! = 720

7! = 5040

8! = 40320

9! = 362880

10! = 3628800

12)Write a TCL script that multiplies the numbers from 1 to 10

Aim: TCL script that multiplies the numbers from 1 to 10

Source Code:

```
# A program that does a loop and prints
# the numbers from 1 to 10
#
set n 10
for {set i 1} {$i <= $n} {incr i} {
    puts $i
}
```

Output:

13)Write a TCL script for Sorting a list using a comparison function

Aim: TCL script for Sorting a list using a comparison function

Source Code:

```
proc compare { mylist } {  
    set len [llength $mylist]  
    set len [expr $len-1]  
  
    for {set i 0} {$i<$len} {incr i} {  
        for {set j 0} {$j<[expr $len-$i]} {incr j} {  
            if { [lindex $mylist $j] > [lindex $mylist [expr $j+1]] } {  
                set temp [lindex $mylist $j]  
                lset mylist $j [lindex $mylist [expr $j+1]]  
                lset mylist [expr $j+1] $temp  
            }  
        }  
    }  
    puts $mylist  
}  
  
set mylist { 7 3 5 2 }  
compare $mylist
```

Output:

2 3 5 7

14 Write a TCL script to (i)create a list (ii) append elements to the list (iii)Traverse the list (iv)Concatenate the list

Create a list

Syntax:

```
set listName { item1 item2 item3 .. itemn }  
# or  
set listName [list item1 item2 item3]  
# or  
set listName [split "items separated by a character" split_character]
```

Source code:

```
#!/usr/bin/tclsh  
  
set colorList1 {red green blue}  
set colorList2 [list red green blue]  
set colorList3 [split "red_green_blue" _]  
puts $colorList1  
puts $colorList2  
puts $colorList3
```

Output:

```
red green blue  
red green blue  
red green blue
```

2) append elements to the list

Syntax:

```
append listName split_character value  
# or  
lappend listName value
```

Source Code:

```
#!/usr/bin/tclsh  
  
set var orange  
append var " " "blue"  
lappend var "red"  
lappend var "green"
```

```
puts $var
```

Output:

```
orange blue red green
```

(iii) Traverse the list and (iv) Concatenate the list

Source Code:

```
set L1 { 1 2 3 }  
puts $L1
```

```
lappend L1 4 5  
puts "After append $L1"
```

```
puts "Traversing list"  
set i 0  
set len [llength $L1]  
while { $i < $len } {  
    puts [lindex $L1 $i]  
    incr i  
}
```

```
set L2 {-1 0}  
puts "List 2 $L2"
```

```
set L3 [concat $L2 $L1]  
puts "After concat $L3"
```

Output:

Output:

```
1 2 3
```

```
After append 1 2 3 4 5
```

```
Traversing list
```

```
1
```

```
2
```

```
3
```

```
4
```

```
5
```

```
List 2 -1 0
```

```
After concat -1 0 1 2 3 4 5
```

15) Write a TCL script to comparing the file modified times.

Source Code:

```
proc newer { file1 file2 } {  
    if ![file exists $file2] {  
        return 1  
    } else {  
        # Assume file1 exists  
        expr [file mtime $file1] > [file mtime $file2]  
    }  
}
```

Output:

16) Write a TCL script to Copy a file and translate to native format.

Source Code:

```
proc File_Copy {src dest} {  
    set in [open $src]  
    set out [open $dest w]  
    puts -nonewline $out [read $in]  
    close $out ; close $in  
}
```

Output:

- 17) a) Write a Perl script to find the largest number among three numbers.**
b) Write a Perl script to print the multiplication tables from 1-10 using subroutines.

a) Write a Perl script to find the largest number among three numbers.

Source Code:

```
#!/usr/bin/perl
print "Enter a number\n";
chomp($a=<stdin>);
print "Enter second number\n";
chomp($b=<stdin>);
print "Enter third number\n";
chomp($c=<stdin>);
$big=0;
$equal=0;
if($a eq $b){
$big = $a;
$equal = $a;
}
elsif($a > $b){
$big=$a;
}
else{
$big = $b;
}
if($equal eq $c){
print "All numbers are same";
}
elsif($big < $c){
$big = $c;
}
else{
print "The biggest number is $big \n";
}
```

Output:

```
Enter a number
789
Enter second number
258
Enter third number
254
The biggest number is 789
```

...Program finished with exit code 0
Press ENTER to exit console.

b) Write a Perl script to print the multiplication tables from 1-10 using subroutines.

Source Code:

```
print "multiplication tables from 1-10 using subroutines\n";
&table(1);
&table(2);
&table(3);
&table(4);
&table(5);
&table(6);
&table(7);
&table(8);
&table(9);
&table(10);

sub table{

    my $i = 1;
    my $loop;

    foreach $loop(@_){
        for($i;$i<=10;$i++){
            my $ans = $i*$loop;
            print"$loop*$i=$ans \n";
        }
        print"\n";
    }
}
```

Output:

multiplication tables from 1-10 using subroutines

```
1*1=1
1*2=2
1*3=3
1*4=4
1*5=5
1*6=6
1*7=7
1*8=8
1*9=9
1*10=10
```

```
2*1=2
2*2=4
2*3=6
```

$2*4=8$
 $2*5=10$
 $2*6=12$
 $2*7=14$
 $2*8=16$
 $2*9=18$
 $2*10=20$

$3*1=3$
 $3*2=6$
 $3*3=9$
 $3*4=12$
 $3*5=15$
 $3*6=18$
 $3*7=21$
 $3*8=24$
 $3*9=27$
 $3*10=30$

$4*1=4$
 $4*2=8$
 $4*3=12$
 $4*4=16$
 $4*5=20$
 $4*6=24$
 $4*7=28$
 $4*8=32$
 $4*9=36$
 $4*10=40$

$5*1=5$
 $5*2=10$
 $5*3=15$
 $5*4=20$
 $5*5=25$
 $5*6=30$
 $5*7=35$
 $5*8=40$
 $5*9=45$
 $5*10=50$

$6*1=6$
 $6*2=12$
 $6*3=18$
 $6*4=24$
 $6*5=30$
 $6*6=36$
 $6*7=42$

$6*8=48$
 $6*9=54$
 $6*10=60$

$7*1=7$
 $7*2=14$
 $7*3=21$
 $7*4=28$
 $7*5=35$
 $7*6=42$
 $7*7=49$
 $7*8=56$
 $7*9=63$
 $7*10=70$

$8*1=8$
 $8*2=16$
 $8*3=24$
 $8*4=32$
 $8*5=40$
 $8*6=48$
 $8*7=56$
 $8*8=64$
 $8*9=72$
 $8*10=80$

$9*1=9$
 $9*2=18$
 $9*3=27$
 $9*4=36$
 $9*5=45$
 $9*6=54$
 $9*7=63$
 $9*8=72$
 $9*9=81$
 $9*10=90$

$10*1=10$
 $10*2=20$
 $10*3=30$
 $10*4=40$
 $10*5=50$
 $10*6=60$
 $10*7=70$
 $10*8=80$
 $10*9=90$
 $10*10=100$

18. Write a Perl program to implement the following list of manipulating functions

- a)Shift
- b)Unshift
- c) Push

a)shiift

shift function

This function returns the first value in an array, removing it and shifting the elements of the array list to the left by one. Shift operation removes the value like pop but is taken from the start of the array instead of the end as in pop. This function returns undef if the array is empty otherwise returns first element of the array.

Syntax: **shift(Array)**

Source Code:

```
#!/usr/bin/perl

# Initializing the array
@x = ('Java', 'C', 'C++');

# Print the Initial array
print "Original array: @x \n";

# Prints the value returned
# by shift function
print "Value returned by shift: ",
      shift(@x);

# Array after shift operation
print "\nUpdated array: @x";
```

Output:

```
Original array: Java C C++
Value returned by shift :Java
Updated array: C C++
```

b)Unshift

unshift function

This function places the given list of elements at the beginning of an array. Thereby shifting all the values in an array by right. Multiple values can be unshift using this operation. This function returns the number of new elements in an array.

Syntax: unshift(Array, List)

Source Code:

```
#!/usr/bin/perl

# Initializing the array
@x = ('Java', 'C', 'C++');

# Print the Initial array
print "Original array: @x \n";

# Prints the number of elements
# returned by unshift
print "No of elements returned by unshift: ",
      unshift(@x, 'PHP', 'JSP');

# Array after unshift operation
print "\nUpdated array: @x";
```

output:

```
Original array: Java C C++
No of elements returned by unshift :5
Updated array: PHP JSP Java C C++
```

c) Push

push function

This function inserts the values given in the list at an end of an array. Multiple values can be inserted separated by comma. This function increases the size of an array. It returns number of elements in new array.

Syntax: push(Array, list)

Source Code:

```
#!/usr/bin/perl
```

```
# Initializing the array
```

```
@x = ('Java', 'C', 'C++');
```

```
# Print the Initial array
```

```
print "Original array: @x \n";
```

```
# Pushing multiple values in the array
```

```
push(@x, 'Python', 'Perl');
```

```
# Printing the array
```

```
print "Updated array: @x";
```

output:

Original array: Java C C++

Updated array: Java C C++ Python Perl

19. a) Write a Perl script to substitute a word, with another word in a string.

b) Write a Perl script to validate IP address and email address.

a) Write a Perl script to substitute a word, with another word in a string.

substitution Operator or 's' operator in [Perl](#) is used to substitute a text of the string with some pattern specified by the user.

Syntax: s/text/pattern

Returns: 0 on failure and number of substitutions on success

Source Code:

```
#!/usr/bin/perl -w
```

```
# String in which text
```

```
# is to be replaced
```

```
$string = "GeeksforGeeks";
```

```
# Use of s operator to replace
```

```
# text with pattern
```

```
$string =~ s/for/to/;
```

```
# Printing the updated string
```

```
print "$string\n";
```

output:

GeekstoGeeks

b) Write a Perl script to validate IP address and email address.

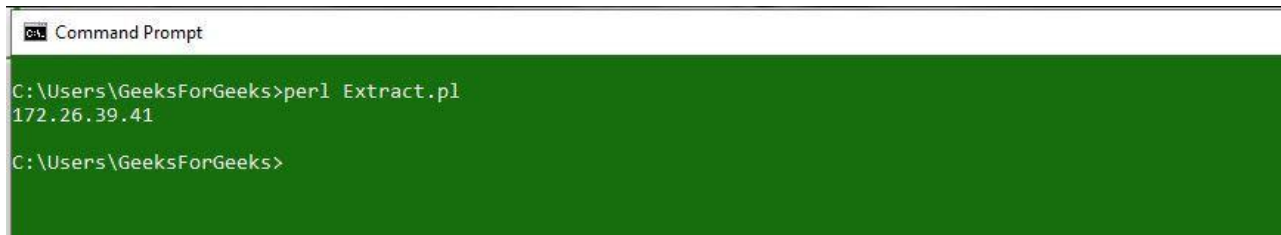
Source Code:

```
#!/usr/bin/perl

my $ip = "MY IP ADDRESS IS172.26.39.41THIS IS A VALID IP ADDRESS";

if($ip =~ /\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}/)
{
    $ip = $1;
    print "$ip\n";
}
```

Output:



```
cmd Command Prompt

C:\Users\GeeksForGeeks>perl Extract.pl
172.26.39.41

C:\Users\GeeksForGeeks>
```

20) Write a Perl script to print the file in reverse order using command line arguments

Source code:

```
#!/usr/bin/perl -w

# Defining string to be reversed
$string = "Hello World";
print scalar reverse($string), "\n";

$string = "Geeks For Geeks";
print scalar reverse($string), "\n";
```

output:

```
dlroW olleH
skeeG roF skeeG
```