Department of CSE

Laboratory Manual	
Course:	B.Tech.
Year & Semester:	III – II
Class:	CSE
Subject:	**Scripting Languages Lab Manual**
Regulation:	R22

BALAJI INSTITUTE OF TECHNOLOGY AND SCIENCE (AUTONOMOUS)

B.Tech (Department of Computer Science & Engineering)

SCRIPTING LANGUAGES LAB

Course Outcomes:

- Ability to understand the differences between Scripting languages and programming Languages.
- Implement logic to solve mathematical and string manipulation problems in Ruby, Perl, TCL.
- Able to Handle File and Data Operations like file handling, list manipulation, and user inputs in scripting languages.
- Apply control structures to perform decision-making and repetitive tasks.
- Enhance Practical Programming Abilities with hands-on experience in scripting for automation, validation, and optimization tasks.

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 users 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 Concatenate the list.
15. Write a TCL script to comparing 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 nonnegative integer

```
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.

```
radius = 0.0
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: 10
The perimeter is 62.83185306.
The area is 314.1592653.
```

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

```
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:
ece
Input your last name:
students
Hello students ece
```

Method II:

```
puts "Input your first name: "
fname = gets.chomp
puts "before change #{fname}"
rev = ''
for i in 1..fname.length
   rev += fname[fname.length - i]
end
puts "after change #{rev}"
puts "Input your last name: "
lname = gets.chomp
puts "before change #{lname}"
rev = ''
for i in 1..lname.length
      rev += lname[lname.length - i]
end
puts "after change #{rev}"
```

Output:

Input your first name:

devender

before change devender

after change redneved

Input your last name:

nayini

before change nayini

after change iniyan

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

```
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

```
puts "enter X value"
x=gets.to_i
puts "enter Y value"
y=gets.to_i
puts "enter Z value"
z=gets.to_i
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:

```
enter X value
10
enter Y value
20
enter Z value
30
z = 30 is greatest.
```

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

```
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 scirpt to check two integers and return true if one of them is 20 otherwise return their sum

```
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

```
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

```
arr= Array["ruby","C++","C#","Java"]
puts arr[0]
print arr
for i in arr
    puts i
end
puts arr
```

Output:

```
ruby
C++
C#
Java
```

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

```
student_marks = Hash.new 0
student_marks['Literature'] = 74
student_marks['Science'] = 89
student_marks['Math'] = 91
total_marks = 0
student_marks.each {|key,value|
        total_marks +=value
    }
puts "Total Marks: "+total_marks.to_s
```

Output:

Total Marks: 254

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

```
set fact 1
for {set i 0} {$i <= 16} {incr i} {
   puts "$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
11! = 39916800
12! = 479001600
13! = 6227020800
14! = 87178291200
15! = 1307674368000
16! = 20922789888000
```

Method II:

```
proc factorial {number} {
  if {$number <= 1} {
    return 1
  }
  return [expr $number * [factorial [expr $number - 1]]]
}
puts [factorial 5]
```

Output:

120

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

```
proc mult {} {
   for {set a 1} {$a <= 10} {incr a} {
      puts "Multiplication table of $a";
      for {set b 1} {$b <= 10} {incr b} {
         puts "$a x $b = [expr $a * $b]"
      }
   }
}
puts [mult];
```

Output:

Multiplication table of 1

1 x 1 = 1

1 x 2 = 2

1 x 3 = 3

1 x 4 = 4

1 x 5 = 5

1 x 6 = 6

1 x 7 = 7

1 x 8 = 8

1 x 9 = 9

1 x 10 = 10

Multiplication table of 2

2 x 1 = 2

2 x 2 = 4

2 x 3 = 6

2 x 4 = 8

2 x 5 = 10

2 x 6 = 12

2 x 7 = 14

2 x 8 = 16

2 x 9 = 18

2 x 10 = 20

Multiplication table of 3

3 x 1 = 3

3 x 2 = 6

3 x 3 = 9

3 x 4 = 12

3 x 5 = 15

3 x 6 = 18

3 x 7 = 21

3 x 8 = 24

3 x 9 = 27

3 x 10 = 30

Multiplication table of 4

4 x 1 = 4

4 x 2 = 8

4 x 3 = 12

4 x 4 = 16

4 x 5 = 20

4 x 6 = 24

4 x 7 = 28

4 x 8 = 32

4 x 9 = 36

4 x 10 = 40

Multiplication table of 5

5 x 1 = 5

5 x 2 = 10

5 x 3 = 15

5 x 4 = 20

5 x 5 = 25

5 x 6 = 30

5 x 7 = 35

5 x 8 = 40

5 x 9 = 45

5 x 10 = 50

Multiplication table of 6

6 x 1 = 6

6 x 2 = 12

6 x 3 = 18

6 x 4 = 24

6 x 5 = 30

6 x 6 = 36

6 x 7 = 42

6 x 8 = 48

6 x 9 = 54

6 x 10 = 60

Multiplication table of 7

7 x 1 = 7

7 x 2 = 14

7 x 3 = 21

7 x 4 = 28

7 x 5 = 35

7 x 6 = 42

7 x 7 = 49

7 x 8 = 56

7 x 9 = 63

7 x 10 = 70

Multiplication table of 8

8 x 1 = 8

8 x 2 = 16

8 x 3 = 24

8 x 4 = 32

8 x 5 = 40

8 x 6 = 48

8 x 7 = 56

8 x 8 = 64

8 x 9 = 72

8 x 10 = 80

Multiplication table of 9

9 x 1 = 9

9 x 2 = 18

9 x 3 = 27

9 x 4 = 36

9 x 5 = 45

9 x 6 = 54

9 x 7 = 63

9 x 8 = 72

9 x 9 = 81

9 x 10 = 90

Multiplication table of 10

10 x 1 = 10

10 x 2 = 20

10 x 3 = 30

10 x 4 = 40

10 x 5 = 50

10 x 6 = 60

10 x 7 = 70

10 x 8 = 80

10 x 9 = 90

10 x 10 = 100

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

```
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

```
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:

```
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.

```
proc newer { file1 file2 } {
  if ![file exists $file2] {
    return 1
  } else {
    # Assume file1 exists
    expr [file mtime $file1] > [file mtime $file2]
     puts "file modification times compared."
  }
}
newer file1.tcl  file2.tcl
```

Output:

file modification times compared

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

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

Output:

File Translated

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

```
    print "enter a value";
$x=<stdin>;
print "enter b value";
$y=<stdin>;
print "enter c value";
$z=<stdin>;
if($a > $b) //if compares string use gt ,lt,le,ge
{
        if($a> $c)
        {
                print " $a is largest number\n";
        }
        else
        {
                print " $c is largest number\n";
        }
}
elsif($b >$c)
{
     print " $b is largest number";
}
else
{
    print " $c is largest nnumber";
}
```

Output:

Enter a value 4

Enter b value 6

Enter c value 5

6 is largest number

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

```
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) POP b) Push c) Shift d) Un shift

POP: The pop function will remove and return the last element of an array

```
my @names = ('cse', 'ece', 'eee');
my $last_one = pop @names;
 print "$last_one\n";
print "@names\n";
```

Output:

eee

cse ece

Push: The push function can add one or more values to the end of an array

```
my @names = ('Foo', 'Bar');
push @names, 'Moo';
print "@names\n";
 my @others = ('Darth', 'Vader');
push @names, @others;
print "@names\n";
```

Output:

Foo Bar Moo

Foo Bar Moo Darth Vader

Shift: If you imagine the array starting on the left-hand side, the shift function will move the whole array one unit to the left. The first element will "fall off" the array and become the function's return value.

```
my @names = ('EEE', 'CSE', 'ECE');
my $first = shift @names;
print "$first\n";
print "@names\n";
```

Output:

EEE

CSE ECE

Un Shift: This is the opposite operation of shift. unshift will take one or more values (or even 0 if that's what you like) and place it at the beginning of the array, moving all the other elements to the right.

```
my @names = ('Foo', 'Bar');
unshift @names, 'Moo';
print "@names\n";
 my @others = ('Darth', 'Vader');
unshift @names, @others;
print "@names\n";
```

Output:

```
Moo Foo Bar
Darth Vader Moo Foo Bar
```

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

```
use strict;
use warnings;
# Define the input string and the words to replace
my $input_string = "The quick brown fox jumps over the lazy dog";
my $word_to_replace = "fox";
my $replacement_word = "cat";
# Perform the substitution
$input_string =~ s/$word_to_replace/$replacement_word/g;
# Print the modified string
print $input_string;
```

Output:

The quick brown cat jumps over the lazy dog

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

```
#!/usr/bin/perl
use strict;
use warnings;
# Define the IP address and email address to validate
my $ip_address = "192.168.0.1";
my $email_address = "example@example.com";
# Validate the IP address
if ($ip_address =~ /^((25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$/) {
    print "Valid IP address: $ip_address\n";
} else {
    print "Invalid IP address: $ip_address\n";
}
# Validate the email address
if ($email_address =~ /^[^\s@]+@[^\s@]+\.[^\s@]+$/) {
    print "Valid email address: $email_address\n";
} else {
    print "Invalid email address: $email_address\n";
}
```

Output:

Valid IP address: 192.168.0.1

Valid email address: example@example.com

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

```
#!/usr/bin/perl
use strict;
use warnings;
my $filename = $ARGV[0];
open my $fh, '<', $filename or die "Could not open file '$filename': $!";
my @lines = reverse <$fh>;
print @lines;
```

Output:

File name devender