MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Sponsored by Chaitanya Bharathi Educational Society, Estd: 1997)

Accredited Six UG Programs 3 times by NBA and NAAC by 'A' Grade, Affiliated to JNTUH, Hyderabad

Department of Computer Science and Engineering

SCRIPTING LANGUAGES LAB MANUAL

III B Tech – II Semester

Branch: **CSE**

Mahatma Gandhi Institute of Technology

Chaitanya Bharathi Post, Gandipet, Hyderabad-500075(A. P.)

INDEX

CONTENT BEYOND SYLLABUS

INTRODUCTION

About Ruby Programming Language

Ruby is a pure Object-Oriented language developed by Yukihiro Matsumoto (also known as Matz in the Ruby community) in the mid 1990's in Japan. Everything in Ruby is an object except the blocks but there are replacements too for it i.e procs and lambda. The objective of Ruby's development was to make it act as a sensible buffer between human programmers and the underlying computing machinery. Ruby has similar syntax to that of many programming languages like C and Java, so it is easy for Java and C programmers to learn. It supports mostly all the platforms like Windows, Mac, Linux.

Ruby is based on many other languages like Perl, Lisp, Smalltalk, Eiffel and Ada. It is an interpreted scripting language which means most of its implementations execute instructions directly and freely, without previously compiling a program into machine-language instructions. Ruby programmers also have access to the powerful RubyGems (RubyGems provides a standard format for Ruby programs and libraries).

Beginning with Ruby programming:

1. Finding a Compiler:

Before starting programming in Ruby, a compiler is needed to compile and run our programs. There are many online compilers that can be used to start Ruby without installing a compiler:

https://www.jdoodle.com/execute-ruby-online

https://repl.it/

There are many compilers available freely for compilation of Ruby programs.

2. Programming in Ruby:

To program in Ruby is easy to learn because of its similar syntax to already widely used languages.

Writing program in Ruby:

Programs can be written in Ruby in any of the widely used text editors like Notepad++, gedit etc. After writing the programs save the file with the extension **.rb**

Advantages of Ruby:

- The code written in Ruby is small, elegant and powerful as it has fewer number of lines of code.
- Ruby allows simple and fast creation of Web application which results in less hard work.
- As Ruby is free of charge that is Ruby is free to copy, use, modify, it allow programmers to make necessary changes as and when required.
- Ruby is a dynamic programming language due to which there is no tough rules on how to built in features and it is very close to spoken languages.

Disadvantages of Ruby:

- Ruby is fairly new and has its own unique coding language which makes it difficult for the programmers to code in it right away but after some practice its easy to use. Many programmers prefer to stick to what they already know and can develop.
- The code written in Ruby is harder to debug, since most of the time it generates at runtime, so it becomes difficult to read while debugging.
- Ruby does not have a plenty of informational resources as compared to other programming languages.
- Ruby is an interpreted scripting language, the scripting languages are usually slower than compiled languages therefore, Ruby is slower than many other languages.

Applications:

- Ruby is used to create web applications of different sorts. It is one of the hot technology at present to create web applications.
- Ruby offers a great feature called Ruby on Rails (RoR). It is a web framework that is used by programmers to speed up the development process and save time.

About PERL Language

Perl is a general purpose, high level interpreted and dynamic programming language. Perl supports both the procedural and Object-Oriented programming. Perl is a lot similar to C syntactically and is easy for the users who have knowledge of C, C++. Since Perl is a lot similar to other widely used languages syntactically, it is easier to code and learn in Perl. Programs can be written in Perl in any of the widely used text editors like Notepad++, gedit, Sbublime etc.

- Perl is a stable, cross platform programming language.
- Though Perl is not officially an acronym but few people used it as **Practical Extraction and Report Language**.
- It is used for mission critical projects in the public and private sectors.
- Perl is an *Open Source* software, licensed under its *Artistic License*, or the *GNU General Public License (GPL)*.
- Perl was created by Larry Wall.
- Perl takes the best features from other languages, such as C, awk, sed, sh, and BASIC, among others.
- Perls database integration interface DBI supports third-party databases including Oracle, Sybase, Postgres, MySQL and others.
- Perl works with HTML, XML, and other mark-up languages.
- Perl supports Unicode.

About **Tool Command Language**

Tcl is shortened form of **Tool Command Language**. John Ousterhout of the University of California, Berkeley, designed it. It is a combination of a scripting language and its own interpreter that gets embedded to the application, we develop with it.

Tcl was developed initially for Unix. It was then ported to Windows, DOS, OS/2, and Mac OSX. Tcl is much similar to other unix shell languages like Bourne Shell (Sh), the C Shell (csh), the Korn Shell (sh), and Perl.

It aims at providing ability for programs to interact with other programs and also for acting as an embeddable interpreter. Even though, the original aim was to enable programs to interact, you can find full-fledged applications written in Tcl/Tk.

Features of Tcl

The features of Tcl are as follows −

- Reduced development time.
- Powerful and simple user interface kit with integration of TK.
- Write once, run anywhere. It runs on Windows, Mac OS X, and almost on every Unix platform.
- Quite easy to get started for experienced programmers; since, the language is so simple that they can learn Tcl in a few hours or days.
- You can easily extend existing applications with Tcl. Also, it is possible to include Tcl in C, C++, or Java to Tcl or vice versa.
- Have a powerful set of networking functions.
- Finally, it's an open source, free, and can be used for commercial applications without any limit.

Applications

Tcl is a general-purpose language and you can find Tcl everywhere. It includes,

- Scalable websites that are often backed by databases.
- High performance web servers build with TclHttpd.
- Tcl with CGI based websites.
- Desktop GUI applications.
- Embedded applications.

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.

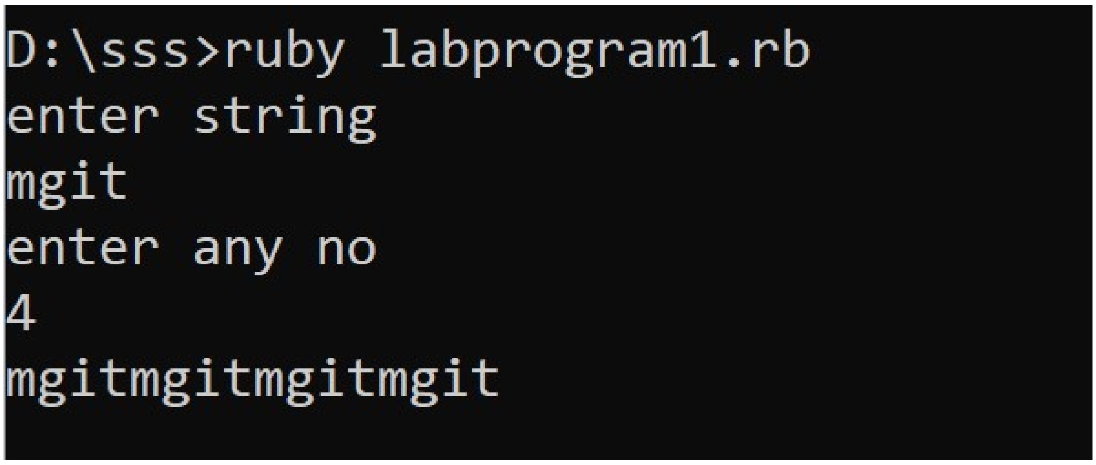
Algorithm

1. Read one string
2. Using sort function we can sort the array element in descending order

Code:

```
puts "enter string"
s1=gets.chomp
puts "enter any no"
no=gets.chomp.to_i
if no > 0
        puts s1 * no
else
        puts "enter +ve no "
end
```

Output:



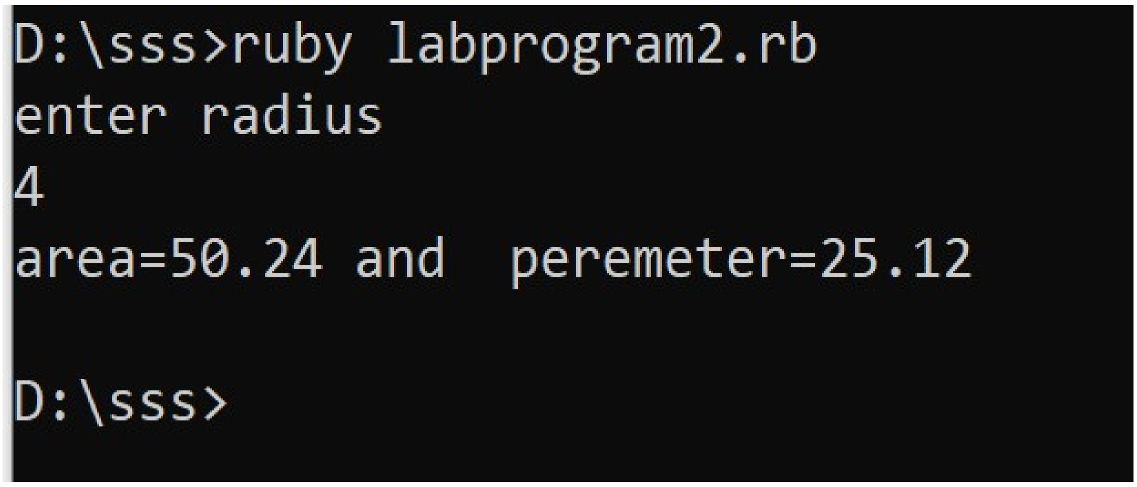
```
D:\sss>ruby labprogram1.rb
enter string
mgit
enter any no
4
mgitmgitmgitmgit
```

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

Code:

```
puts "enter radius"
r=gets.chomp.to_f
pi=3.14
area=pi * r * r
peremeter= 2 * pi * r
puts "area=#{area} and  peremeter=#{peremeter}"
```

Output:



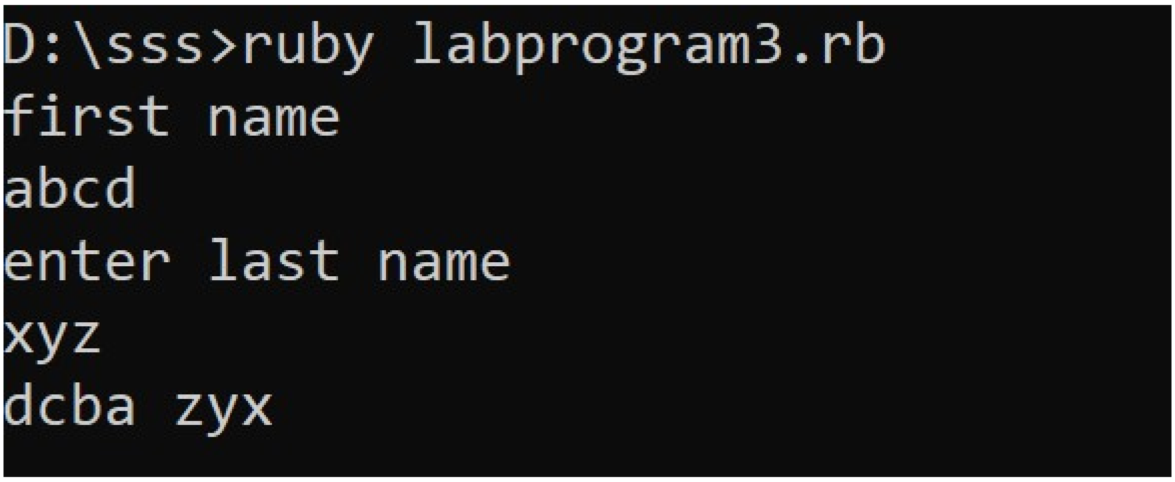
```
D:\sss>ruby labprogram2.rb
enter radius
4
area=50.24 and  peremeter=25.12

D:\sss>
```

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.

Code:

```
puts "first name"
fname=gets.chomp
puts "enter last name"
lname=gets.chomp
rfn=fname.reverse
rln=lname.reverse
puts "#{fname.reverse} #{rln}"
```

Output:



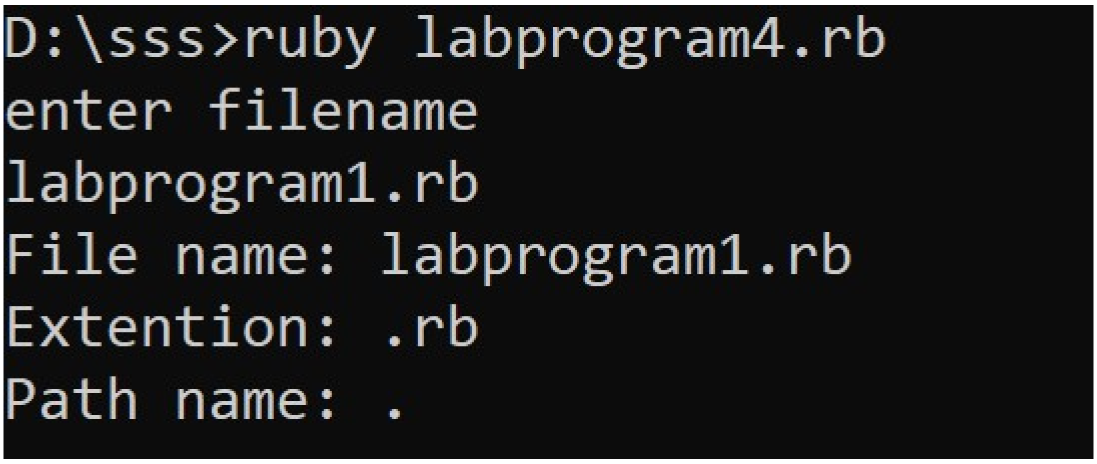
```
D:\sss>ruby labprogram3.rb
first name
abcd
enter last name
xyz
dcba zyx
```

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

Code:

```
puts "enter filename"
file = gets.chomp

fbname = File.basename (file) # file name
puts "File name: "+fbname

ffextn = File.extname  (file)   # file extention
puts "Extention: "+ffextn

path_name= File.dirname  (file) # path name
puts "Path name: "+path_name
```

Output:



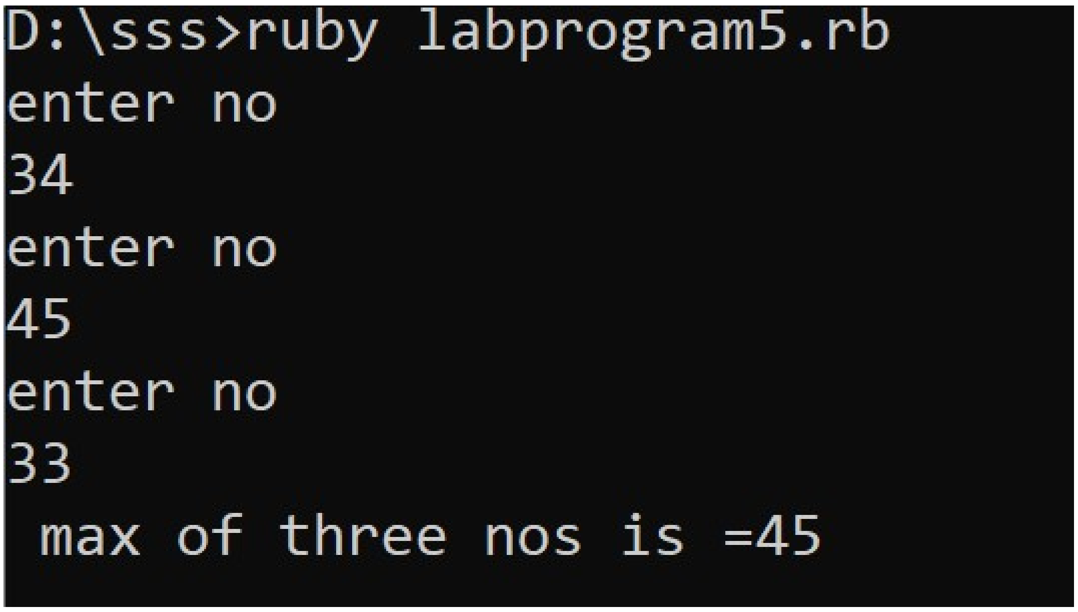
```
D:\sss>ruby labprogram4.rb
enter filename
labprogram1.rb
File name: labprogram1.rb
Extention: .rb
Path name: .
```

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

Code:

```
puts "enter no";
p =gets.chomp.to_i;
puts "enter no";
q =gets.chomp.to_i;
puts "enter no";
r =gets.chomp.to_i;
max = p > q ? (p > r ? p : r) : (r > q ? r : q) ;
puts " max of three nos is =#{max}";
```

Output:



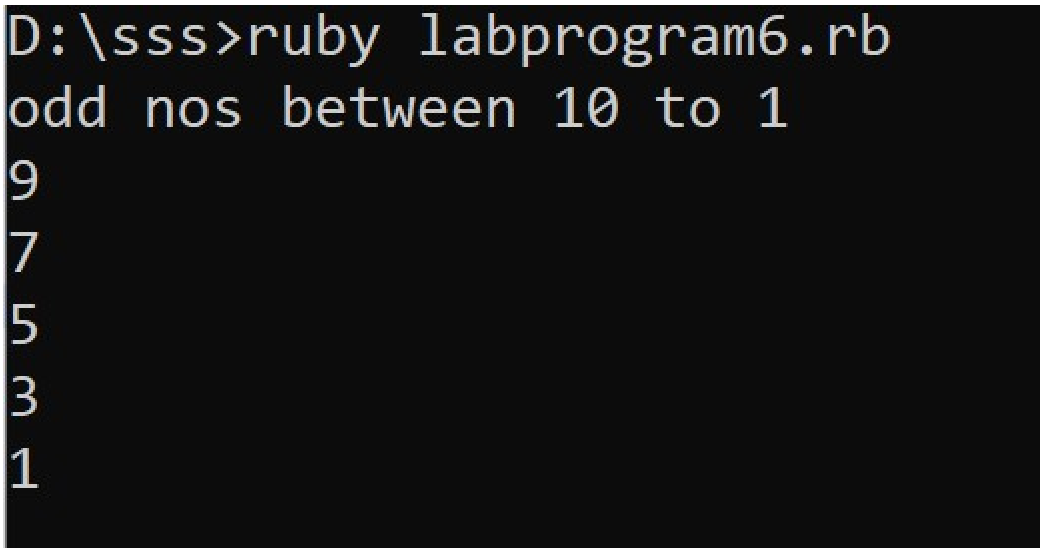
```
D:\sss>ruby labprogram5.rb
enter no
34
enter no
45
enter no
33
 max of three nos is =45
```

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

Code:

```
x=10
puts "odd nos between 10 to 1"
while x > 0
        if x % 2 != 0
                puts "#{x}"
        end
        x= x-1
end
```

Output:



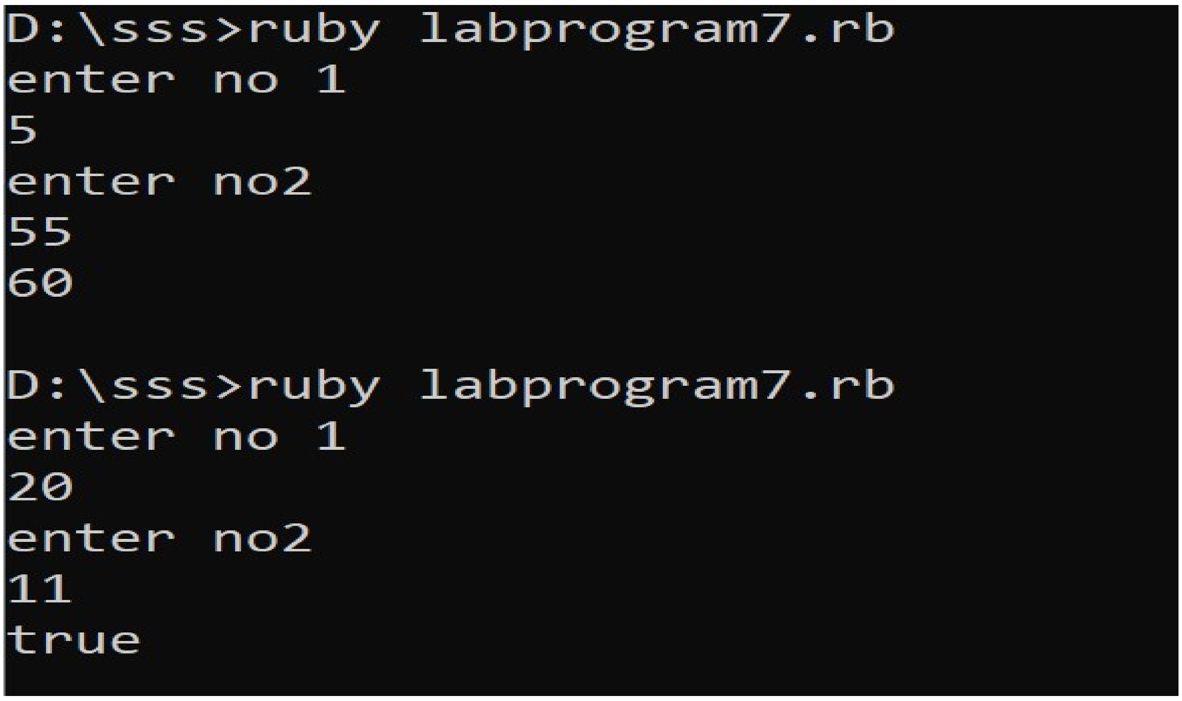
```
D:\sss>ruby labprogram6.rb
odd nos between 10 to 1
9
7
5
3
1
```

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

Code:

```
def check(a,b)
        if(a==20 || b==20)
                return true
        else
                return a+b
        end
end
puts "enter no 1"
x=gets.chomp.to_i
puts "enter no2"
y=gets.chomp.to_i
res=check(x,y)
puts res
```

Output:



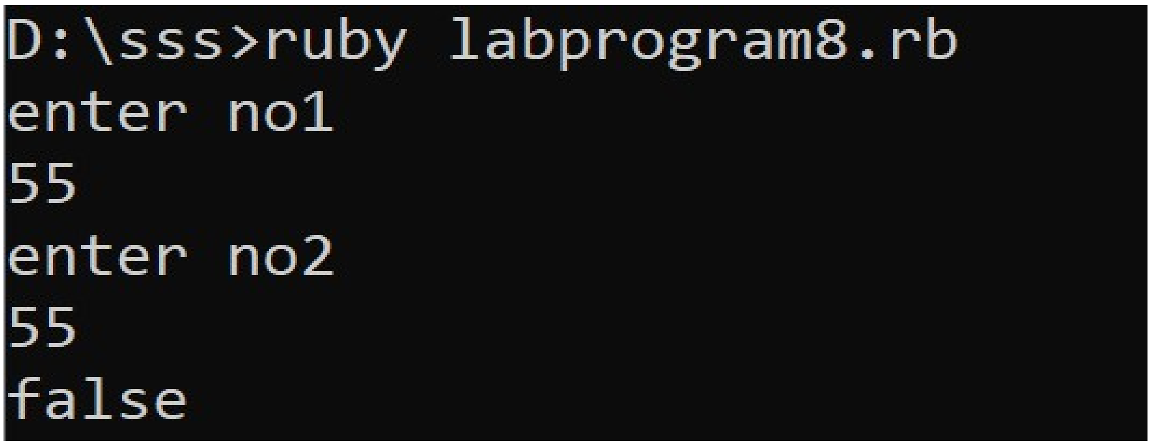
```
D:\sss>ruby labprogram7.rb
enter no 1
5
enter no2
55
60

D:\sss>ruby labprogram7.rb
enter no 1
20
enter no2
11
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.

Code:

```
def check(p,q)
        if((p<0 && q>100) or (p>100 && q<0))
                return true
        else
                return false
        end


end
puts "enter no1"
a=gets.chomp.to_i
puts "enter no2"
b=gets.chomp.to_i
puts check(a,b)
```

Output:



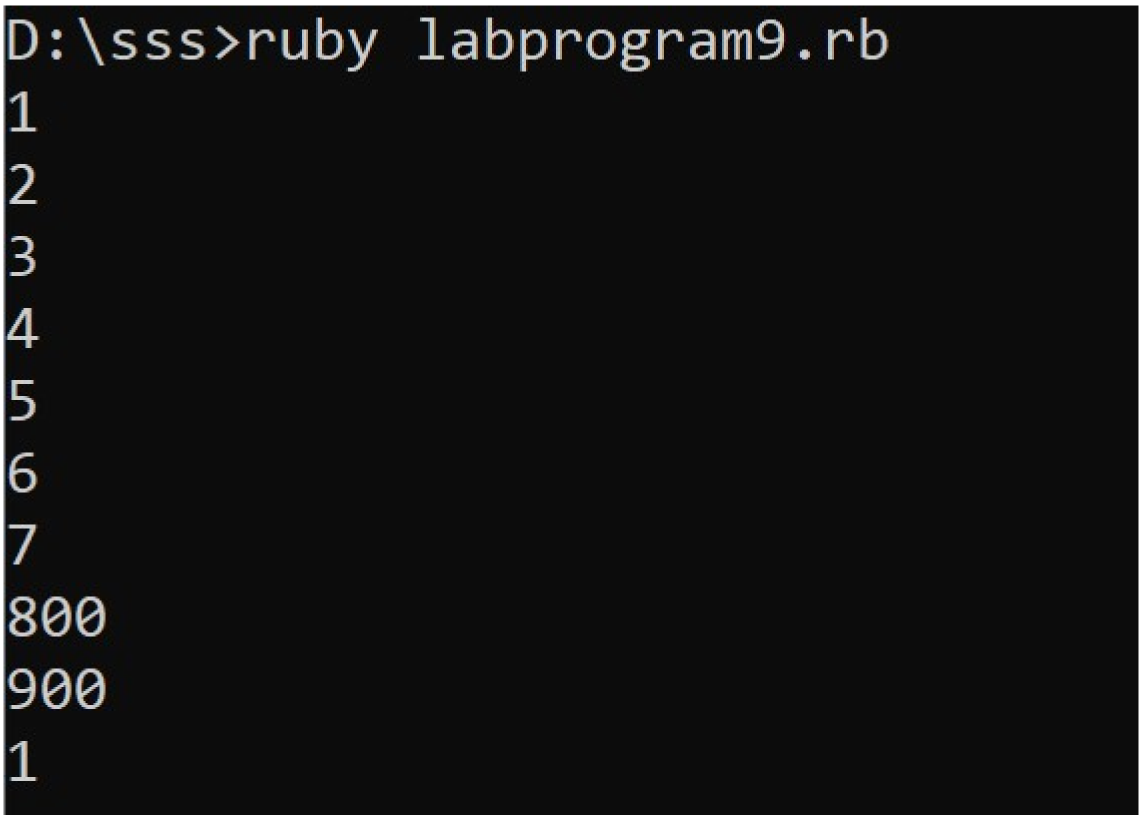
```
D:\sss>ruby labprogram8.rb
enter no1
55
enter no2
55
false
```

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

Code:

```
arr1=[1,2,3,4,5,6,7,800,900]
puts arr1
#arr1.push(999) adding elements at end fo the array
#puts arr1
#puts arr1[4] display certian element
#puts arr1[-1] display the elements in reverse dirction
#puts arr1[3,5] diplay the elements form certian position and no of elements
#puts arr1.at(5) know the element at the given pos
#puts arr1.first(3) display first three elements
#puts arr1.last(5) display last five elements
#puts arr1.sort sort the array
#puts arr1.sort.reverse stort the array in reverse order
#x=[1,2,3]
#y=[4,5,6]
#z=x+y adding two arrays
#puts z
#puts arr1.delete(3) delete the specified elemet
#puts arr1
#arr1.pop delete the last element
#puts arr1
#puts arr1.length know the size of arry
#puts arr1.size know the size of arry
#arr1 << 10 adding the elements at the end
#puts arr1
#arr1.unshift(33) adding the elements at the end
#puts arr1
#arr1.insert(2,44)adding the eelenmntes at certian pos
#puts arr1

puts arr1.shift #delete the elements first elemet from array
```

Output:



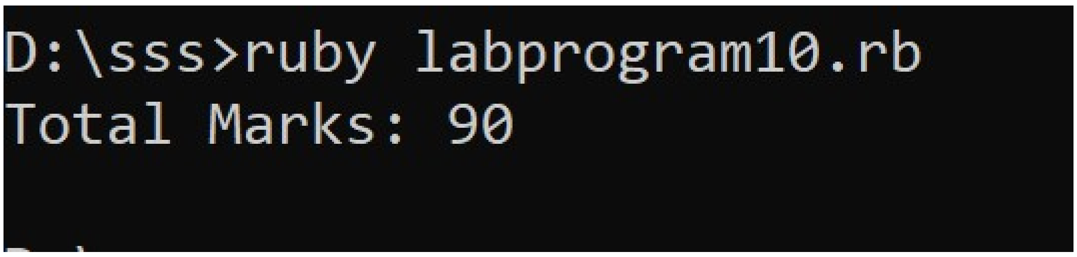
```
D:\sss>ruby labprogram9.rb
1
2
3
4
5
6
7
800
900
1
```

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

    Code:

    ```
    marks = Hash.new
    marks['c'] = 30
    marks['java'] = 30
    marks['sl'] = 30
    marks['ml'] = 0

    tmarks = 0
    marks.each do |key,value|
            tmarks +=value
        end
    puts "Total Marks: "+tmarks.to_s
    ```

    Ouput:



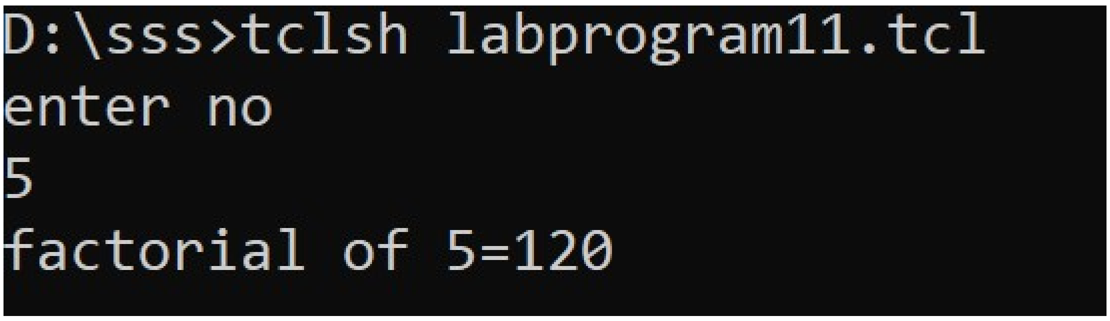
```
D:\sss>ruby labprogram10.rb
Total Marks: 90
```

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

    Code:

```
set i 1;
set product 1;
puts "enter no";
gets stdin x;# if i given x=4

#set x 5;
  while {$i <= $x} {
    set product [expr $product * $i];
    incr i;
  }
puts "factorial of $x=$product";
```

Output:



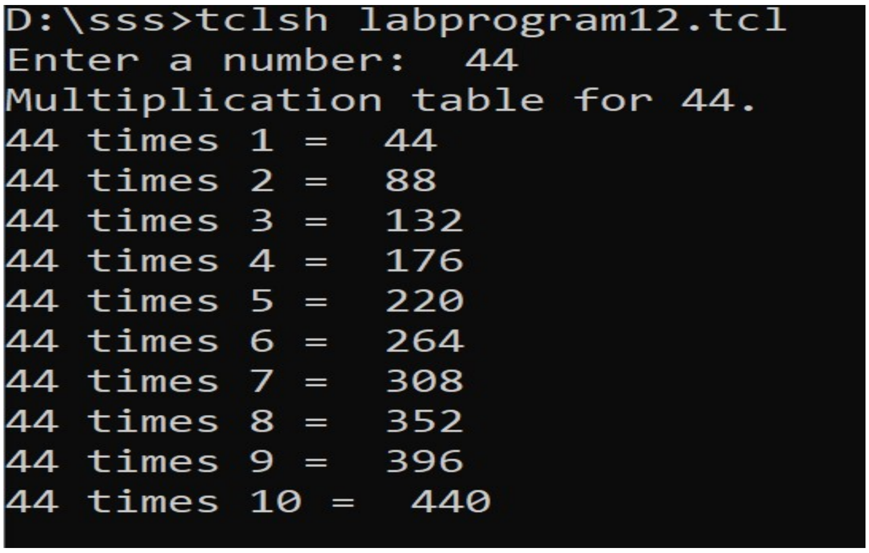
```
D:\sss>tclsh labprogram11.tcl
enter no
5
factorial of 5=120
```

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

Code:

```
proc times_table { x }
{
  puts "Multiplication table for $x."
  for {set i 1 } { $i <= 10} {incr i } {
    set answer [expr $x * $i]
    puts "$x times $i =  $answer"
  }
}

proc run_table { } {
  puts -nonewline "Enter a number:  "
  flush stdout
  gets stdin x
  times_table $x
}
run_table
#end of program
```

Output:



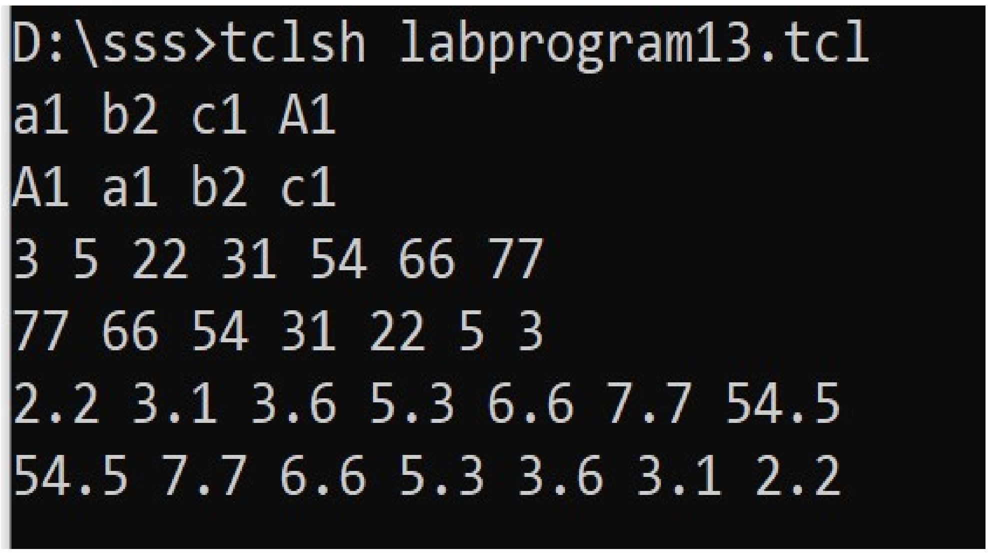
```
D:\sss>tclsh labprogram12.tcl
Enter a number:  44
Multiplication table for 44.
44 times 1 =  44
44 times 2 =  88
44 times 3 =  132
44 times 4 =  176
44 times 5 =  220
44 times 6 =  264
44 times 7 =  308
44 times 8 =  352
44 times 9 =  396
44 times 10 =  440
```

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

Code:

```
set lst1 {a1 b2 c1 A1};
puts $lst1;
#lsort {a1 b2 c1 A1};
set lst2 [lsort -ascii $lst1];
puts $lst2;
set lst3 {5 54 3 22 66 77 31};
set slst3 [lsort -integer $lst3];
puts $slst3;
set rslst3 [lsort -integer -decreasing $lst3];
puts $rslst3;
set lst4 {5.3 54.5 3.6 2.2 6.6 7.7 3.1};
set slst4 [lsort -real $lst4];
puts $slst4;
set ralst4 [lsort -real -decreasing $lst4 ];
puts $ralst4;
```

Output:



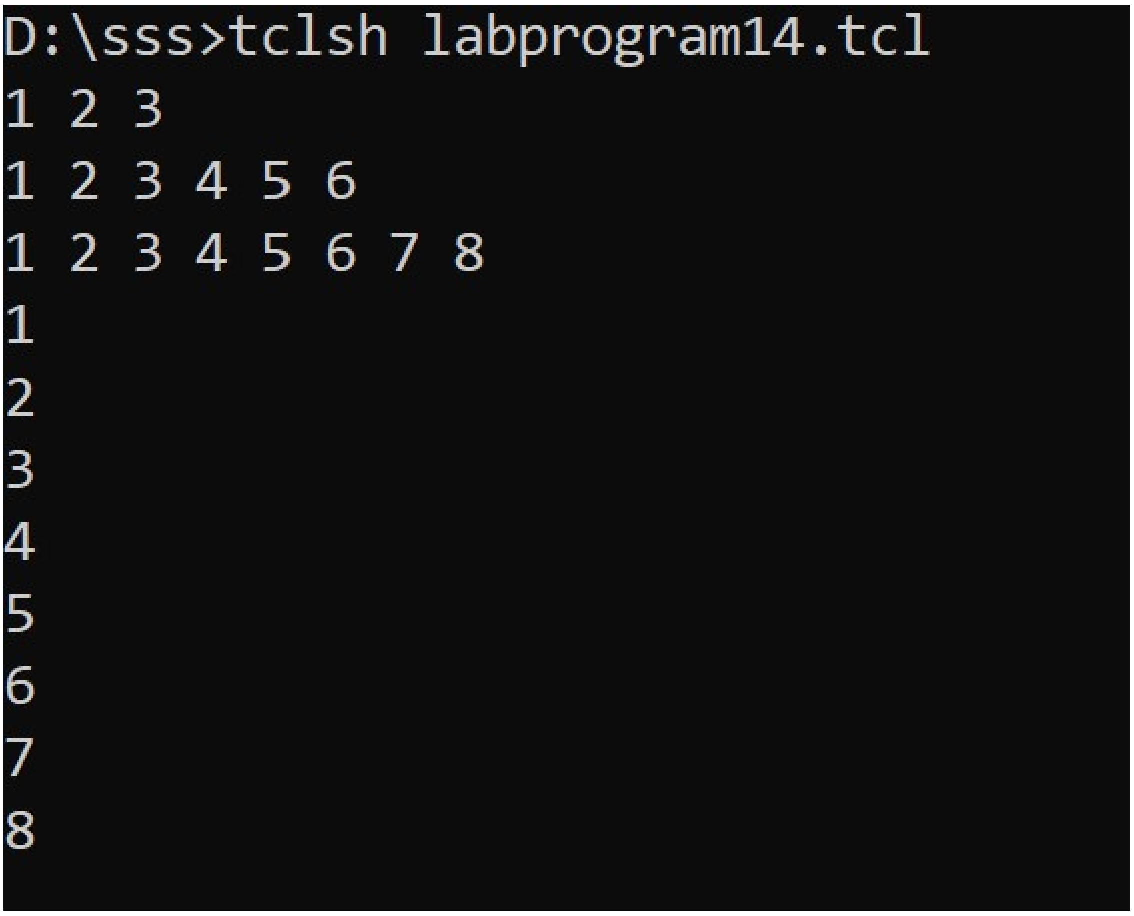
```
D:\sss>tclsh labprogram13.tcl
a1 b2 c1 A1
A1 a1 b2 c1
3 5 22 31 54 66 77
77 66 54 31 22 5 3
2.2 3.1 3.6 5.3 6.6 7.7 54.5
54.5 7.7 6.6 5.3 3.6 3.1 2.2
```

14. Write a TCL script to
    (i) Create a list
    (ii) Append elements to the list
    (iii) Traverse the list
    (iv) Concatenate the list

Code:

```
#createing list
set lst1 {1 2 3}
puts $lst1
set lst2 {4 5 6}
#concat list
set lst3 [concat $lst1 $lst2]
puts  $lst3;
#append list
lappend lst3 7 8
puts  $lst3;
#travse list
foreach val $lst3 {
puts $val;
}
```

Output:



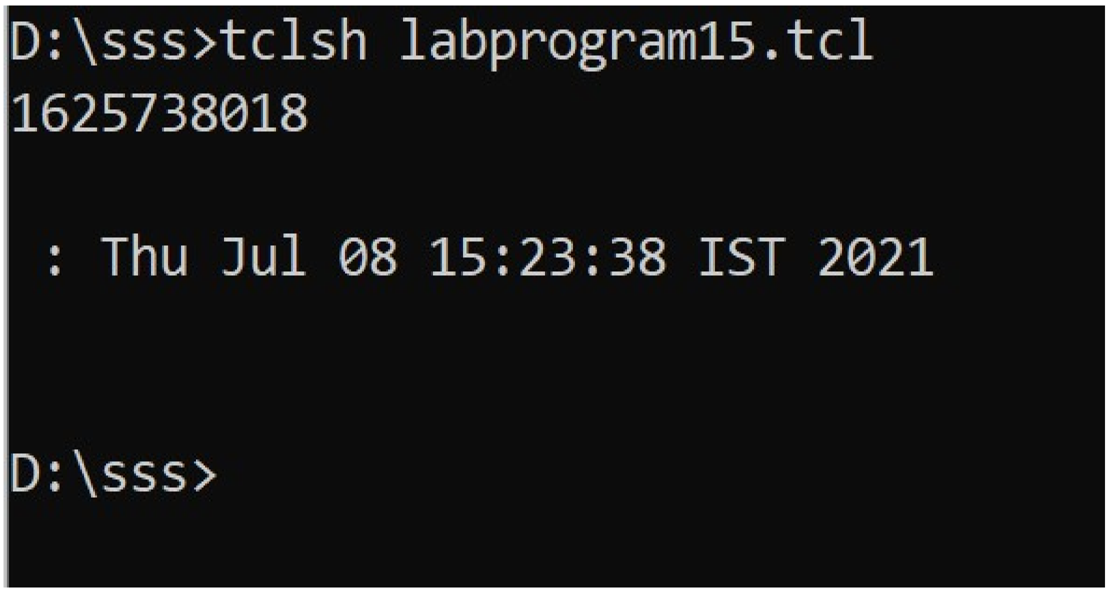
```
D:\sss>tclsh labprogram14.tcl
1 2 3
1 2 3 4 5 6
1 2 3 4 5 6 7 8
1
2
3
4
5
6
7
8
```

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

    Code:

```
# Get the modification time:
set now [clock seconds]
set filename "filemodifiedtime1.tcl"
set timestamp [file mtime $filename]
puts $timestamp
# Set the modification time to 'now':
#file mtime $filename [clock seconds]
#uts $mtime
puts "\n : [clock format $timestamp]\n"
```

    Output:



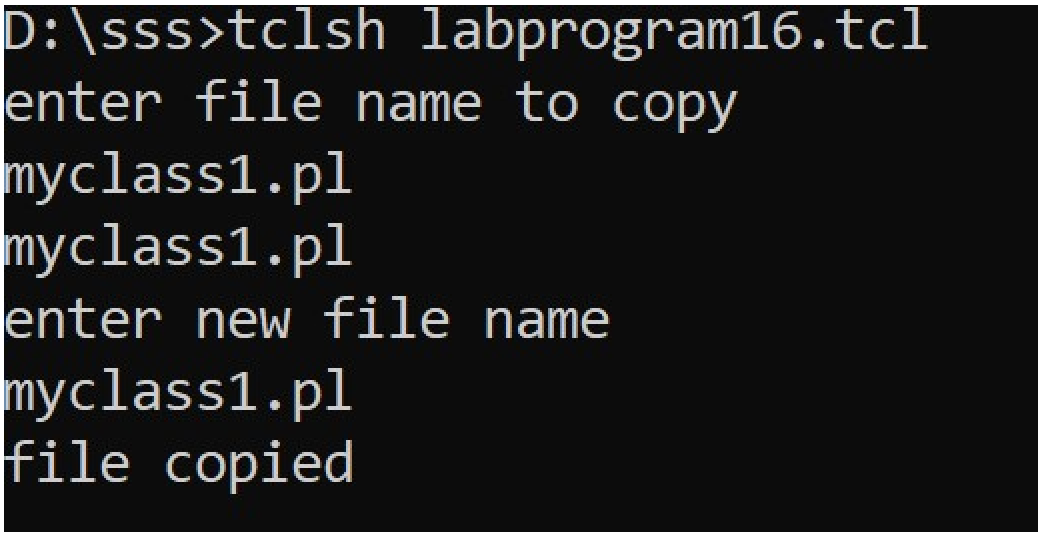
```
D:\sss>tclsh labprogram15.tcl
1625738018

 : Thu Jul 08 15:23:38 IST 2021


D:\sss>
```

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

Code:

```
puts "enter file name to copy";
gets stdin fn
puts $fn
set fp [open $fn r]
set data [read $fp]
#puts $file_data
#close $fp

puts "enter new file name";
gets stdin fn1
set fp1 [open $fn1 w+]
#puts "enter data to write";
#gets stdin data;
puts $fp1 $data;
close $fp1
puts "file copied"
```

Output:



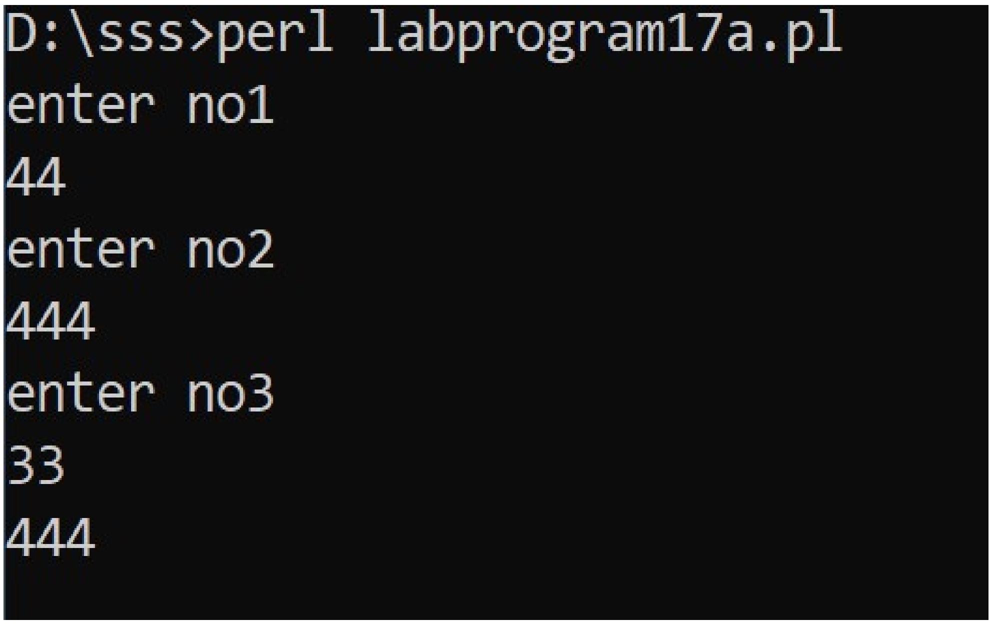
```
D:\sss>tclsh labprogram16.tcl
enter file name to copy
myclass1.pl
myclass1.pl
enter new file name
myclass1.pl
file copied
```

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

Code:

```
#!/usr/bin/perl
 #use strict;
 use warnings;
 print "enter no1\n";
 $p=<STDIN>;
 print "enter no2\n";
 $q=<STDIN>;
 print "enter no3\n";
 $r=<STDIN>;

 $max = $p > $q ? ($p > $r ? $p : $r) : ($r > $q ? $r : $q) ;
 # implement all arithmetic operators
 print $max;
 exit(0);
```

Otput:



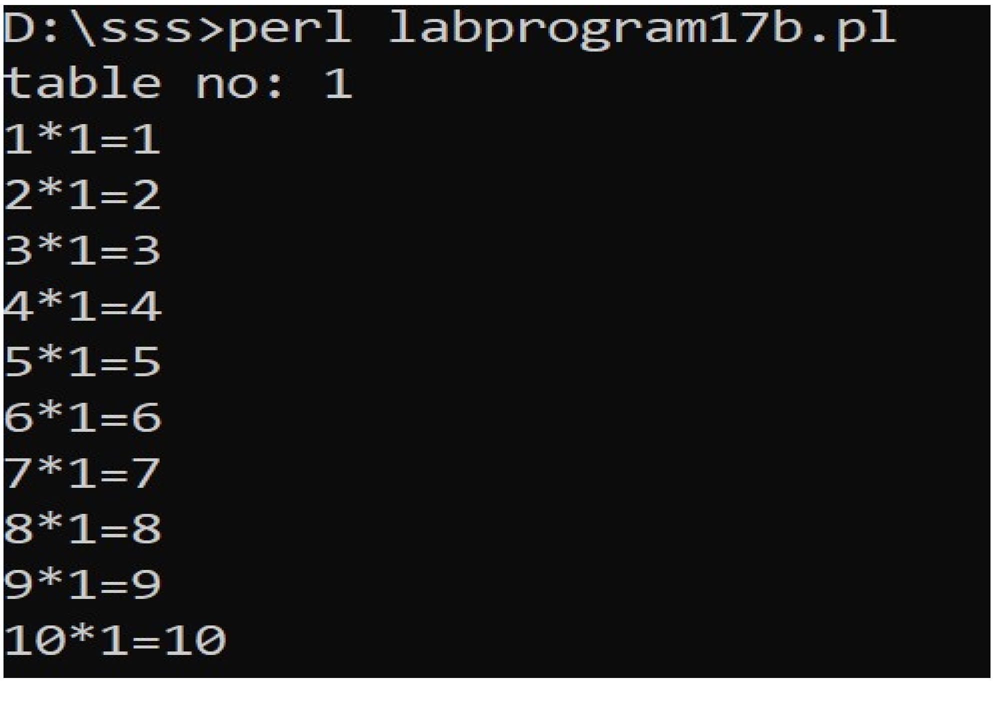
```
D:\sss>perl labprogram17a.pl
enter no1
44
enter no2
444
enter no3
33
444
```

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

Code:

```
sub multable
{
        $n=@_[0];
        print("table no: ".$n."\n");
        my $i=1;
        my $re=0;
        while($i<=10)
        {
                $re=$i*$n;
                print "$i*$n=$re\n";
                $i++;
        }
        print "------------\n";
}

$i=1;
        while($i<=10)
        {
                multable($i);
                $i++;
        }
```

Output:



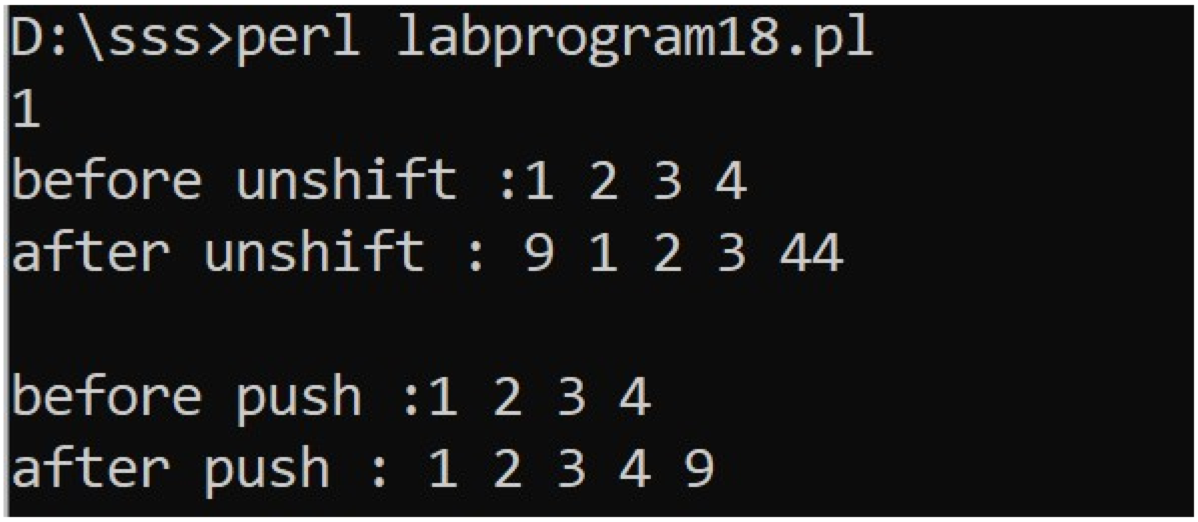
```
D:\sss>perl labprogram17b.pl
table no: 1
1*1=1
2*1=2
3*1=3
4*1=4
5*1=5
6*1=6
7*1=7
8*1=8
9*1=9
10*1=10
```

18. Write a Perl script to print the multiplication tables from 1-10 using subroutines.
    a) Shift b) Unshift c) Push

Code:

```
#implement shift,unshift,push,pop methods
#implement shift
@arr1=(1,2,3,4);
$ele=shift @arr1;
# shift delete the first element in the array and returns the element

print $ele;
#implement unshift
@arr1=(1,2,3,4);
#unshift add the element in the array at the begiening
print "before unshift :@arr1";
unshift (@arr1,9);
print "after unshift : @arr1";

#implement poo
@arr1=(1,2,3,4);
$ele=pop @arr1;
# pop delete the last element in the array and returns the element

print $ele."\n";
#implement push
@arr1=(1,2,3,4);
#push add the element in the array at the last pos..
print "before push :@arr1";
push (@arr1,9);
print "after push : @arr1";
```

Output:



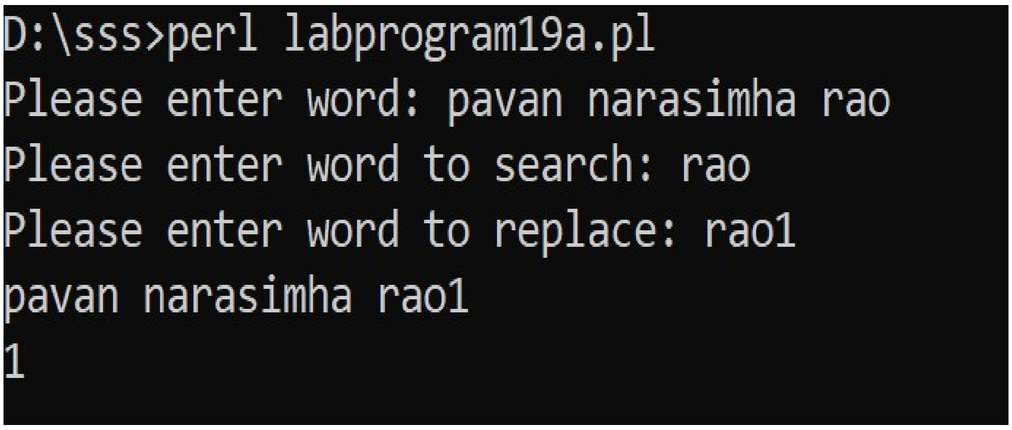
```
D:\sss>perl labprogram18.pl
1
before unshift :1 2 3 4
after unshift : 9 1 2 3 44

before push :1 2 3 4
after push : 1 2 3 4 9
```

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

Code:

```
use strict;
use warnings;
my $myString = "";

print "Please enter word: ";
$myString = <STDIN>;
chomp($myString);

print "Please enter word to search: ";
my $myWord2= <STDIN>;
chomp($myWord2);


print "Please enter word to replace: ";
my $myWord3 =<STDIN>;
chomp($myWord3);

my $myCount = $myString =~ s/$myWord2/$myWord3/;
print "$myString \n";
print "$myCount \n";
exit(0);
```

Ouput:



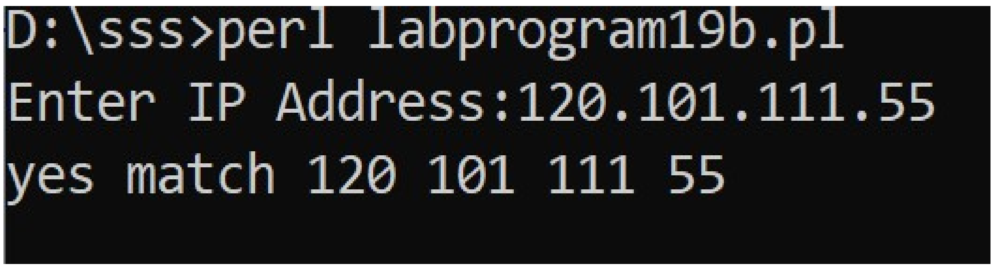
```
D:\sss>perl labprogram19a.pl
Please enter word: pavan narasimha rao
Please enter word to search: rao
Please enter word to replace: rao1
pavan narasimha rao1
1
```

b).Write a Perl script to validate IP address

Code:

```
print "enter ip address";
$ip=<STDIN>;
if(($ip =~ /^(\d{1,3})\.(\d{1,3})\.(\d{1,3})\.(\d{1,3})$/) && ($1 <= 255 && $2
<= 255 && $3 <= 255 && $4 <= 255 )){
print "ip address is $ip" ;
}
else
{
print "wrong ip address";
}
```

Output:



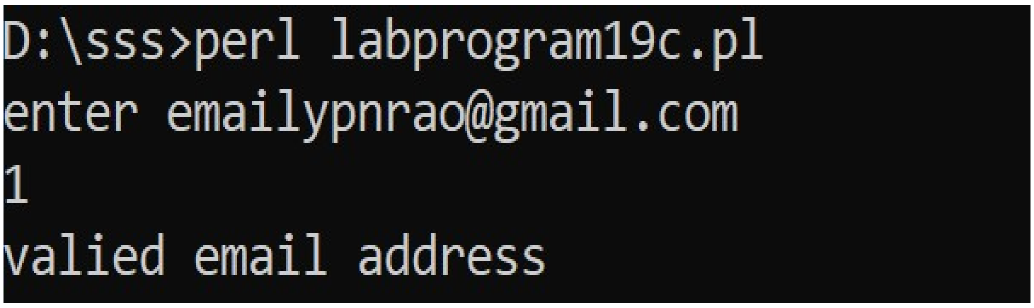
```
D:\sss>perl labprogram19b.pl
Enter IP Address:120.101.111.55
yes match 120 101 111 55
```

c). Write a Perl script to validate email address.

Code:

```
use strict;
use warnings;
use 5.010;

use Email::Address;
print "enter email";
my $line = <STDIN> ;
chomp($line);

my $addresses = Email::Address->parse($line);
print $addresses,"\n";
if($addresses==1)
{
print "valied email address";
}
else{
print "invalied email address";
}
```

Output:



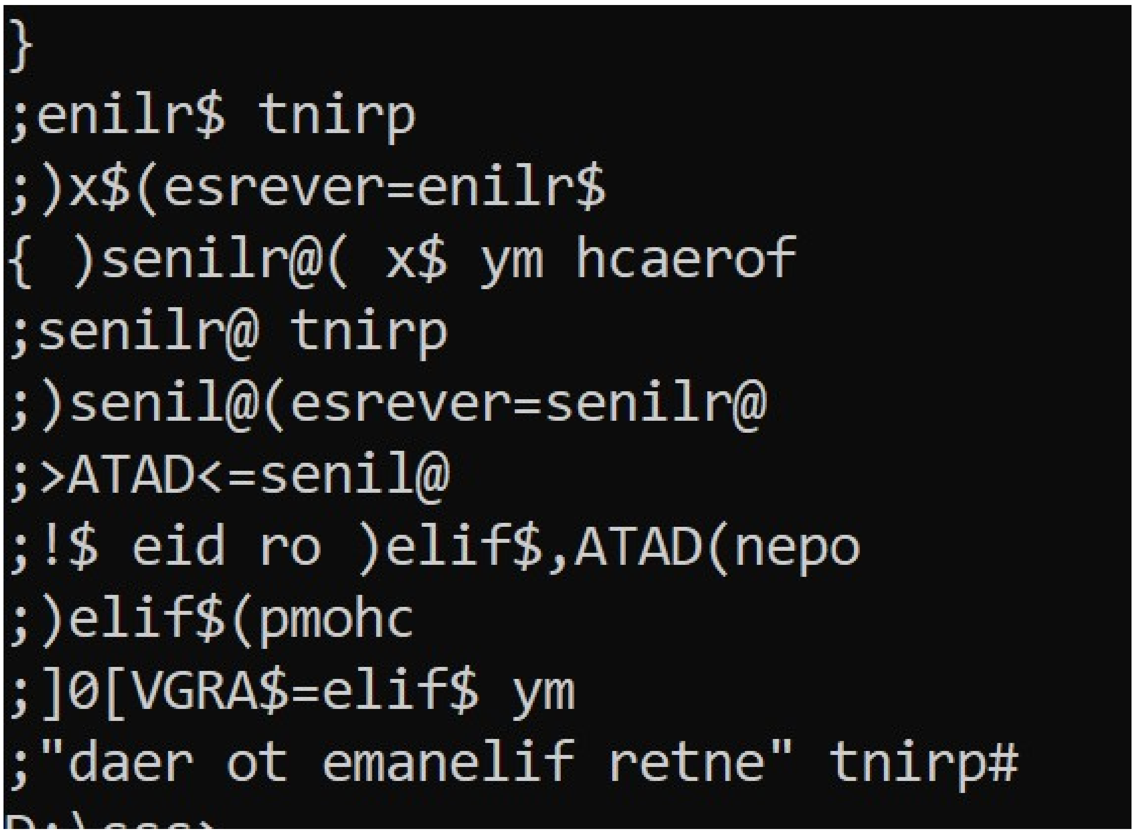
```
D:\sss>perl labprogram19c.pl
enter emailypnrao@gmail.com
1
valied email address
```

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

Code:

```
#print "enter filename to read";
my $file=$ARGV[0];
chomp($file);
open(DATA,$file) or die $!;
@lines=<DATA>;
@rlines=reverse(@lines);
print @rlines;
foreach my $x (@rlines) {
  $rline=reverse($x);
  print $rline;
}
close(DATA);
```

Output:



```
}
;enilr$ tnirp
;)x$(esrever=enilr$
{ )senilr@( x$ ym hcaerof
;senilr@ tnirp
;)senil@(esrever=senilr@
;>ATAD<=senil@
;!$ eid ro )elif$,ATAD(nepo
;)elif$(pmohc
;]0[VGRA$=elif$ ym
;"daer ot emanelif retne" tnirp#
```

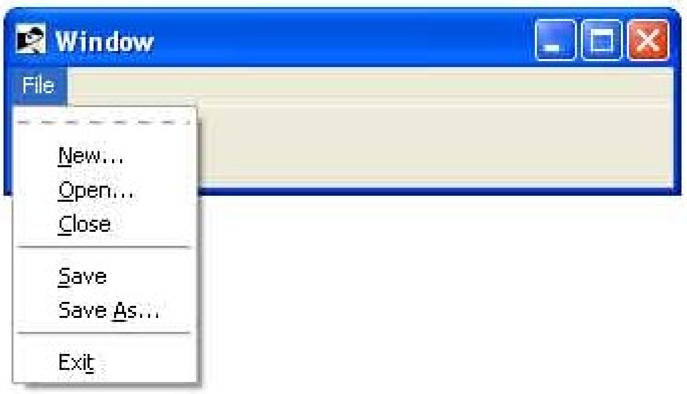
CONTENT BEYOND SYLLABUS

1. Write a Ruby Script to demonstrate Menu Driven Application

Code:

```
require "tk"

root = TkRoot.new
root.title = "Window"

menu_click = Proc.new {
  Tk.messageBox(
    'type'    => "ok",
    'icon'    => "info",
    'title'   => "Title",
    'message' => "Message"
  )
}

file_menu = TkMenu.new(root)

file_menu.add('command',
          'label'     => "New...",
          'command'   => menu_click,
          'underline' => 0)
file_menu.add('command',
          'label'     => "Open...",
          'command'   => menu_click,
          'underline' => 0)
file_menu.add('command',
          'label'     => "Close",
          'command'   => menu_click,
          'underline' => 0)
file_menu.add('separator')
file_menu.add('command',
          'label'     => "Save",
          'command'   => menu_click,
          'underline' => 0)
file_menu.add('command',
          'label'     => "Save As...",
          'command'   => menu_click,
            'underline' => 5)
file_menu.add('separator')
file_menu.add('command',
            'label'     => "Exit",
            'command'   => menu_click,
            'underline' => 3)

menu_bar = TkMenu.new
menu_bar.add('cascade',
           'menu'  => file_menu,
           'label' => "File")

root.menu(menu_bar)
Tk.mainloop
```

Ouput:



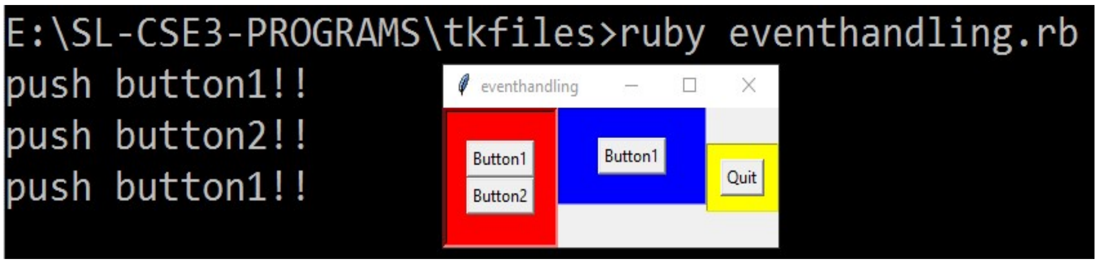
2. Implement Ruby TK application to demonstrate Event Handling

Code:

```
require "tk"

f1 = TkFrame.new {
  relief 'sunken'
  borderwidth 3
  background "red"
  padx 15
  pady 20
  pack('side' => 'left')
}
f2 = TkFrame.new {
  relief 'groove'
  borderwidth 1
  background "yellow"
  padx 10
  pady 10
  pack('side' => 'right')
}

f3 = TkFrame.new {
  relief 'groove'
  borderwidth 1
  background "blue"
  padx 30
  pady 20
  pack('side' => 'top')
}

TkButton.new(f3) {
  text 'Button1'
  command {print "push button1!!\n"}
  pack('fill' => 'x')
}

TkButton.new(f1) {
  text 'Button1'
  command {print "push button1!!\n"}
  pack('fill' => 'x')
}
```

```
TkButton.new(f1) {
  text 'Button2'
  command {print "push button2!!\n"}
  pack('fill' => 'x')
}
TkButton.new(f2) {
  text 'Quit'
  command 'exit'
  pack('fill' => 'x')
}
Tk.mainloop
```

Output:



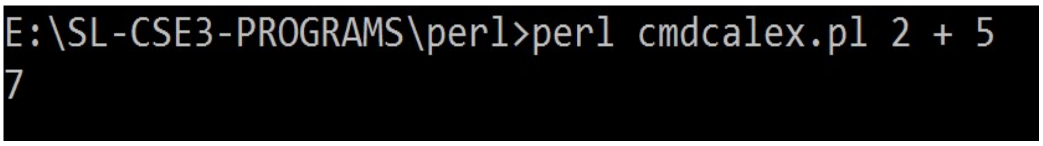
```
E:\SL-CSE3-PROGRAMS\tkfiles>ruby eventhandling.rb
push button1!!
push button2!!
push button1!!
```

3. Write a PERL script to demonstrate Command Line Arguments

   Code:

```
#!/usr/bin/perl -w
if ($#ARGV != 2 ) {
        print "usage: mycal number1 op number2\neg: mycal 5 + 3 OR mycal 5 -
2\n";
        exit;
}
$n1=$ARGV[0];
$op=$ARGV[1];
$n2=$ARGV[2];
$ans=0;
if ( $op eq "+" ) {
        $ans = $n1 + $n2;
}
elsif ( $op eq "-"){
        $ans = $n1 - $n2;
}
elsif ( $op eq "/"){
        $ans = $n1 / $n2;
}
elsif ( $op eq "*"){
        $ans = $n1 * $n2;
}
else {
        print "Error: op must be +, -, *, / only\n";
        exit;
}
print "$ans\n";
```

   Output:



```
E:\SL-CSE3-PROGRAMS\perl>perl cmdcalex.pl 2 + 5
7
```

4. Write a PERL script to demonstrate to Packages, Modules and Classes

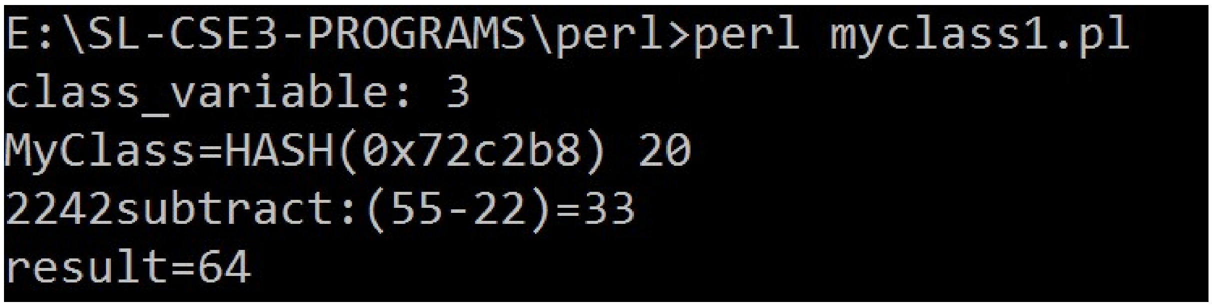
Code:
MyClass.pm

```
package MyClass;
use strict;
use warnings;

# class variable ('our' for package visibility)
#
our $a1 = 3;  # Would like to bind to a variable
our $class_variable2 = 3;
our $c=0;
sub new {
     my $class = shift;
     my $self = { };
     bless $self, $class;
     return $self;
}

sub add {
    my $self = shift;
    print "class_variable: $a1\n";
    #$a=shift;
    #$b=shift;
   ($a,$b)=@_;
   print $self;
    print " $a\n";
    print "$b";
    print $a+$b;

    #print "class_variable: $class_variable2\n";
    #++$class_variable; # prove that other instances will see this change
}
sub subtract
{
my $self=shift;
$a=shift;
$b=shift;
 #$c=$a-$b;
print "subtract:($a-$b)=";
print $a-$b;
}
sub mul
{
my $self=shift;
$a=shift;
$b=shift;
$c=$a*$b;
return $c;

}
1;
```

Myclass1.pl

```
#!/usr/bin/perl

use strict;
use warnings;
use MyClass;
my $res;
my $foo = MyClass->new();
$foo->add(20,22);
$foo->subtract(55,22);
$res=$foo->mul(8,8);
print "\nresult=$res ";
```

Ouput:



```
E:\SL-CSE3-PROGRAMS\perl>perl myclass1.pl
class_variable: 3
MyClass=HASH(0x72c2b8) 20
2242subtract:(55-22)=33
result=64
```

Ruby Viva Questions

1. What is Ruby programming language?
2. Who is the developer of Ruby?
3. Why Ruby is known as language flexibility?
4. List some features of Ruby?
5. Explain some differences between Ruby and Python
6. Name some operators used in Ruby.
7. What is RubyGems in Ruby programming language?
8. What is RubyGems in Ruby Programming Language?
9. What are Ruby variables.
10. What is the difference between nil and false in Ruby?
11. Expain Ruby data types.
12. What is the use of load and require in Ruby?
13. Expalin Ruby if-else statement.
14. Expalin Case statement in Ruby.
15. Explain For loop in Ruby.
16. Explain While loop in Ruby.
17. Explain Until loop in Ruby.
18. Explain Break loop in Ruby.
19. Explain Next loop in Ruby.
20. Explain Redo loop in Ruby.
21. Explain Retry loop in Ruby.
22. How will you comment in Ruby?
23. Explain Retry Object.
24. How to create Ruby Object?

PERL Viva Questions

1. What is Perl?
2. What are the features of Perl programming?
3. What are the benefits of Perl programming in using it in web based applications?
4. Is perl a case sensitive language?
5. What is a perl identifier?
6. What are data types that perl supports?
7. What are scalar data types in perl?
8. What are Arrays in perl?
9. What are Hashes in perl?
10. How will you declare a variable in perl?
11. What is variable context in perl?
12. What is scalar context?
13. What is a list context?
14. What is boolean context?
15. What is void context?
16. What is interpolative context?
17. What is the difference between single quoted string and double quoted string?
18. What is the purpose of _FILE_ literal?
19. What is the purpose of _LINE_ literal?
20. What is the purpose of _PACKAGE_ literal?
21. How will you access an element of a perl array?
22. What is range operator?
23. How will you get size of an array?
24. How will you add an element to an end of an array?
25. How will you add an element to the beginning of an array?
26. How will you remove an element from end of an array?
27. How will you remove an element from begining of an array?
28. How will you get slice from an array?
29. How will you get replace elements of an array?
30. How will you convert a string to an array?
31. How will you convert an array to string?
32. How will you sort an array?
33. What is the purpose of $[variable?
34. How will you merge two arrays?
35. How will you create Hashes in perl?
36. How will you get element from Hashes in perl?
37. How will you get all keys from Hashes in perl?
38. How will you get all values from Hashes in perl?
39. How will you check if key exists in a hash or not?
40. How will you get the size of hash?
41. How will you add an element to a hash?
42. How will you remove an element from a hash?
43. What is the purpose of next statement?

44. What is the purpose of last statement?
45. What is the purpose of continue statement?
46. What is the purpose of redo statement?
47. What is the purpose of goto Label statement?
48. What is the purpose of goto Expr statement?
49. What is the purpose of goto &NAME statement?
50. What is the purpose of ** operator?
51. What is the purpose of <=> operator?
52. What is the purpose of lt operator?
53. What is the purpose of gt operator?
54. What is the purpose of le operator?
55. What is the purpose of ge operator?
56. What is the purpose of eq operator?
57. What is the purpose of ne operator?
58. What is the purpose of cmp operator?
59. What is the purpose of **= operator?
60. What is the purpose of q{ } operator?
61. What is the purpose of qq{ } operator?
62. What is the purpose of qx{ } operator?
63. What is the purpose of . operator?
64. What is the purpose of x operator?
65. What is the purpose of .. operator?
66. What is the purpose of ++ operator?
67. What is the purpose of −− operator?
68. What is the purpose of −> operator?
69. What is the purpose of localtime() function?
70. What is the purpose of gmtime() function?
71. What is the difference between localtime() and gmtime() functions?
72. What is the purpose of time() function?
73. What is the purpose of strftime() function?
74. How will you define a subroutine in perl?
75. How will you call a subroutine in perl?
76. How will you access the parameters passed to a perl subroutine?
77. How will you get the count of parameters passed to a perl subroutine?
78. What is the purpose of my operator?
79. What is the default scope of perl variables?
80. What are lexical variables in perl?
81. What is purpose of local operator in perl?
82. What is dynamic scoping?
83. What is lexical scoping?
84. What are state variables in perl?
85. What is Subroutine Call Context?
86. What is a Perl references?
87. How will you create a reference for a variable?

88. How will you create a reference for a array?
89. How will you create a reference for a hash?
90. How will you create a reference for a subrouting?
91. What is dereferencing?
92. How will you dereference a reference?
93. What is circular reference?
94. How will you open a file in read-only mode?
95. How will you open a file in writing mode?
96. How will you open a file in writing mode without truncating it?
97. What is the purpose of close() function?
98. What is the purpose of getc() function?
99. What is the purpose of read() function?

TCL VIVA QUESTIONS

1. What Is Tcl?
2. How To Increment Eacl Element In A List?
3. How To Run A Package In Tcl ?
4. How to create arrays in Tcl?
5. How Increment A Character?
6. How Do You Find The Length Of A String Without Using String Length Command In Tcl?
7. How To Check Whether A String Is Palindrome Or Not Using Tcl Script?
8. What is namespace?
9. Upvar Command?
10. How to create procedures in TCL?
11. What is the purpose of eval command explain?
12. What is the purpose of source command explain?
13. What is the purpose of exec command explain?