KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

SCRIPTING LANGUAGES LABORATORY

Subject Code :KG23ACD319

Regulation : KGR23

AcademicYear :2025-2026

III B. TECH II SEMESTER

COMPUTER SCIENCE AND ENGINEERING

KGREDDYCOLLEGEOFENGINEERINGANDTECHNOLOGY

Affiliated by JNTUH,Chilkur,(V),Moinabad(M)R.RDist,TS-501504

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

VISION AND MISSION OF THE INSTITUTION

VISION:

To become an institution which is internationally recognized for its holistic approach to engineering, innovative teaching and learning culture, research and entrepreneurial ecosystem, And sustainable social impact in the community.

MISSION:

• To offer undergraduate and post-graduate programs which are supported through industry relevant curriculum and innovative teaching and learning processes that would help students succeed in their professional careers.
• To provide faculty and students with an ecosystem that fosters innovation,research, entrepreneurship, and international exposure through strategic partnerships with government organizations and collaboration with industries.
• To provide holistic learning environment to students which will contribute to their personal and Professional growth and enable them to become leaders in the irrespective fields.
• To contribute to the development of the region by using our technological expertise to work with nearby communities and support them in their social and economic development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

VISION:

To be recognized as a department of excellence by stimulating a learning environment in which students and faculty will thriveand grow to achieve their professional,institution a land Societal goals.

MISSION:

• To provide high quality technical education to students that will enable life-long learning and Build expertise in advanced technologies in Computer Science and Engineering.
• To promote research and development by providing opportunities to solve complex engineering problems in collaboration with industry and government agencies.
• To encourage professional development of students that will in culcate ethical values and leadership skills through entrepreneurship while working with the community to address societalissues.

PROGRAM EDUCATIONAL OBJECTIVES

PEO1: Graduates will provide solutions to difficult and challenging issues in their profession by applying computer science and engineering theory and principles.
PEO2: Graduate shave successful careers in computer science and engineering fields or will be able to successfully pursue advanced degrees.
PEO3: Graduates will communicate effectively ,work collaboratively and exhibit high levels of professionalism, moral and ethical responsibility.
PEO4: Graduates will develop the ability to understand and analyze engineering issues in a broader perspective with ethical responsibility towards sustainable development.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM OUTCOMES

POI: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
POII: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems, reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
POIII: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental factors.
PO IV: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of information to provide valid conclusions.
POV: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling, to complex engineering activities with an understanding of limitations. .
POVI: The engineer and society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal, and cultural issues and the responsibilities relevant to professional engineering practice. .
POVII: Environment and sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of and need for sustainable development.
POVIII: Ethics: Apply ethical principles and commit to professional ethics, responsibilities, and norms of engineering practice.
POIX: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams and multidisciplinary settings.
POX: Communication: Communicate effectively on complex engineering activities with the engineering community and society at large, including writing reports, preparing design documentation, making presentations, and giving and receiving clear instructions. .
POXI: Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these as a member and leader in a team to manage projects in multidisciplinary environments.
PO XII: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the context of technological change

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PROGRAM SPECIFIC OUTCOMES

PSO1: Computer Science and Engineering graduates will be able to analyze, design, develop, and test intelligent systems using mathematical foundations and management principles, and apply computational solutions to develop secure applications and hardware prototypes.
PSO2: Graduates will be able to analyze contemporary research issues in various areas of Computer Science and Engineering, identify research gaps, and conduct research in specialized or emerging fields.
PSO3: Graduates will develop skills to solve problems in programming concepts, evaluate environmental and social issues with ethical responsibility, and manage projects in multidisciplinary fields, fostering careers, entrepreneurship, and higher studies.

KG REDDY College of Engineering & Technology
AN AUTONOMOUS INSTITUTION

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCRIPTING LANGUAGES LABORATORY

Course Code: KG23ACD319 **LTPC**

B.Tech. III Year II Sem **0 0 2 1**

Course Objectives: The objectives of this course for the students are to:

1. Understand the concepts of scripting languages for developing web based projects
2. Understand the applications the of Ruby, TCL, Perl scripting languages
3. Develop the ability to apply control structures, functions, and modules
4. Familiarize students with scripting tools and libraries
5. Enhance debugging, testing, and execution skills

Course Outcomes:

1. **CO 1:** Ability to understand the differences between scripting languages and programming languages(K4)
2. **CO 2:** Gain some fluency programming in Ruby, Perl, TCL(K2)
3. **CO 3:** Develop and debug small to medium Python / Perl / JavaScript programs. (K6)
4. **CO 4:** Work with files, regular expressions, and data structures in scripting languages.(K5)
5. **CO 5:** scripting knowledge in real-world applications like wh development or system administration(K3)

List of Experiments

S.N0	Name of The Experiment
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 luser 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	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 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 numb b) Write a Perl script to print the multiplication tables from 1-10 subroutines.
18	18. Write a Perl program to implement the following list of manipul functions a) Shift b) Unshift c)Push
19	a) Write a Perl script to substitute a word, with another word na string. b) Write a Peri script to validate IP address and email address.
20	Write a Perl script to print the file in reverse order using command line arguments

TEXTBOOK:

1. The World of Scripting Languages, David Barron, Wiley Publications,

2. Ruby Programming language by David Flanagan and Yukihiro Matsumoto O'Reilly

3. "Programming Ruby" The Pragmatic Programmers guide by Dabve Thomas Second edition

REFERENCE BOOKS:

1. Open Source Web Development with LAMP using Linux Apache MySQL, Perl and PHP, J.Lee and B. Ware (Addison Wesley) Pearson Education.

2. Perl by Example, E. Quigley, Pearson Education.

3. Programming Perl, Larry Wall, T. Christiansen and J. Orwant, O'Reil SPD.

4. Tcl and the Tk Tool kit, Ousterhout, Pearson Education.

5. Perl Power, J. P. Flynt, Cengage Learning.

EXPERIMENT-1

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

CODE:

```
# Function to repeat a string n times
def repeat_string(str, n)
if n < 0
return "Invalid input! n must be non-negative."
end

return str * n
end

# User input
print "Enter a string: "
input_str = gets.chomp

print "Enter a non-negative integer: "
n = gets.to_i

# Function call
result = repeat_string(input_str, n)

# Output
puts "Resulting string: #{result}"
```

OUTPUT:

```
Enter a string: Hello
Enter a non-negative integer: 4
Resulting string: HelloHelloHelloHello
```

EXPERIMENT 2

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

CODE:

```
# Program to calculate area and perimeter of a circle

# Taking input from user
print "Enter the radius of the circle: "
radius = gets.to_f

# Check for valid input
if radius < 0
  puts "Invalid input! Radius cannot be negative."
else
  # Calculations
  perimeter = 2 * Math::PI * radius
  area = Math::PI * radius * radius

  # Display results
  puts "Perimeter (Circumference) of the circle: #{perimeter.round(2)}"
  puts "Area of the circle: #{area.round(2)}"
end
```

OUTPUT:

```
Enter the radius of the circle: 5
Perimeter (Circumference) of the circle: 31.42
Area of the circle: 78.54
```

EXPERIMENT 3

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

CODE:

```
# Program to reverse first and last name

# Taking input from user
print "Enter your first name: "
first_name = gets.chomp

print "Enter your last name: "
last_name = gets.chomp

# Reversing order
full_name = last_name + " " + first_name

# Display result
puts "Name in reverse order: #{full_name}"
```

OUTPUT:

```
Enter your first name: John
Enter your last name: Smith
Name in reverse order: Smith John
```

EXPERIMENT 4

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

CODE:

```
# Program to find the extension of a file

# Taking input from user
print "Enter the filename: "
filename = gets.chomp

# Extracting extension
if filename.include?(".")
  extension = filename.split(".").last
puts "File extension is: #{extension}"
else
  puts "No extension found in the filename."
End
```

OUTPUT:

```
Enter the filename: example.txt
File extension is: txt
```

EXPERIMENT 5

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

CODE:

```
# Program to find the greatest of three numbers

# Taking input from user
print "Enter first number: "
a = gets.to_f

print "Enter second number: "
b = gets.to_f

print "Enter third number: "
c = gets.to_f

# Finding the greatest number
if a >= b && a >= c
  greatest = a
elsif b >= a && b >= c
  greatest = b
else
  greatest = c
end

# Display result
puts "The greatest number is: #{greatest}"
```

OUTPUT:

```
Enter first number: 10
Enter second number: 25
Enter third number: 15
The greatest number is: 25
```

EXPERIMENT 6

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

CODE:

```
# Program to print odd numbers from 10 to 1

for i in 10.downto(1)
  if i % 2 != 0
    puts i
  end
end
```

OUTPUT:

```
9
7
5
3
1
```

EXPERIMENT 7

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

CODE:

```
# Program to check two integers

# Taking input from user
print "Enter first number: "
a = gets.to_i

print "Enter second number: "
b = gets.to_i

# Condition check
if a == 20 || b == 20
result = true
else
  result = a + b
end

# Display result
puts "Result: #{result}"
```

OUTPUT:

Case 1

```
Enter first number: 20
Enter second number: 5
Result: true
```

Case 2

```
Enter first number: 10
Enter second number: 5
Result: 15
```

EXPERIMENT 8

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:

```
# Program to check two temperatures

# Taking input from user
print "Enter first temperature: "
t1 = gets.to_f

print "Enter second temperature: "
t2 = gets.to_f

# Condition check
if (t1 < 0 && t2 > 100) || (t2 < 0 && t1 > 100)
  result = true
else
  result = false
end

# Display result
puts "Result: #{result}"
```

OUTPUT:

Case 1

```
Enter first temperature: -5
Enter second temperature: 120 Result: true
```

Case 2

```
Enter first temperature: 20
Enter second temperature: 50
Result: false
```

EXPERIMENT 9

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

CODE:

```
# Program to print elements of an array

# Define an array
arr = [10, 20, 30, 40, 50]

# Printing elements
puts "Array elements are:"
for i in arr
  puts i
end
```

OUTPUT:

```
Array elements are:
10
20
30
40
50
```

EXPERIMENT 10

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

CODE:

```
# Program to calculate total marks from a hash

# Sample hash: subjects as keys, marks as values
student marks = {
  "Math" => 85,
  "English" => 75,
  "Science" => 90,
  "History" => 80
}

# Calculate total marks
total_marks = student_marks.values.sum

# Display result
puts "Total marks of the student: #{total_marks}"
```

OUTPUT:

```
Total marks of the student: 330
```

EXPERIMENT-11

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

CODE:

```
# Factorial program in TCL

# Function to calculate factorial
proc factorial {n} {
    if {$n == 0 || $n == 1} {
        return 1
    } else {
        return [expr {$n * [factorial [expr {$n - 1}]]}]
    }
}

# Input from user
puts -nonewline "Enter a number: "
flush stdout
gets stdin num

# Calculate factorial
set result [factorial $num]

# Output result
puts "Factorial of $num is $result"
```

OUTPUT:

```
Enter a number: 5
Factorial of 5 is 120
```

EXPERIMENT-12

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

CODE:

```
# Multiply numbers from 1 to 10

set result 1

for {set i 1} {$i <= 10} {incr i} {
    set result [expr {$result * $i}]
}

puts "Product of numbers from 1 to 10 is $result"
```

OUTPUT:

```
Product of numbers from 1 to 10 is 3628800
```

EXPERIMENT-13

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

CODE:

```
# Custom comparison function (for ascending order)
proc compareNumbers {a b} {
    if {$a < $b} {
        return -1
    } elseif {$a > $b} {
        return 1
    } else {
        return 0
    }
}
# Define a list
set myList {5 2 9 1 7 3}
# Sort the list using the comparison function
set sortedList [lsort -command compareNumbers $myList]
# Output the result
puts "Original List: $myList"
puts "Sorted List: $sortedList"
```

OUTPUT:

```
Original List: 5 2 9 1 7 3
Sorted List: 1 2 3 5 7 9
```

EXPERIMENT-14

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:

```
# (i) Create a list
set list1 {1 2 3}
puts "Initial List1: $list1"

# (ii) Append elements to the list
lappend list1 4 5 6
puts "After appending elements to List1: $list1"

# (iii) Traverse the list
puts "Traversing List1:"
foreach item $list1 {
    puts $item
}

# Create another list for concatenation
set list2 {7 8 9}
puts "List2: $list2"

# (iv) Concatenate the lists
```

```
set finalList [concat $list1 $list2]

puts "Concatenated List: $finalList"
```

OUTPUT:

```
Initial List1: 1 2 3

After appending elements to List1: 1 2 3 4 5 6

Traversing List1:

1

2

3

4

5

6

List2: 7 8 9

Concatenated List: 1 2 3 4 5 6 7 8 9
```

EXPERIMENT-15

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

CODE:

```
# TCL script to compare file modification times

# Input file names
puts -nonewline "Enter first file name: "
flush stdout
gets stdin file1

puts -nonewline "Enter second file name: "
flush stdout
gets stdin file2

# Check if files exist
if {![file exists $file1] || ![file exists $file2]} {
    puts "One or both files do not exist."
    exit
}

# Get modification times
set time1 [file mtime $file1]
set time2 [file mtime $file2]

# Compare modification times
if {$time1 > $time2} {
```

```
    puts "$file1 is more recently modified than $file2"
} elseif {$time1 < $time2} {
    puts "$file2 is more recently modified than $file1"
} else {
    puts "Both files were modified at the same time"
}
```

OUTPUT:

```
Enter first file name: file1.txt
Enter second file name: file2.txt
file2.txt is more recently modified than file1.txt
```

EXPERIMENT-16

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

CODE:

```
# Source and destination file names
set source_file "input.txt"
set destination_file "output.txt"

# Open source file in read mode
set in [open $source_file r]

# Read entire content
set data [read $in]

# Close input file
close $in

# --- Translation Step ---
# Convert Windows line endings (\r\n) to Unix format (\n)
regsub -all {\r\n} $data "\n" translated_data

# (Optional) Convert tabs to spaces (example of another translation)
regsub -all {\t} $translated_data "    " translated_data

# Open destination file in write mode
set out [open $destination_file w]
```

```
# Write translated content
puts $out $translated_data

# Close output file
close $out

puts "File copied and translated successfully!"
```

OUTPUT:

```
Hello World!
This is TCL.
```

EXPERIMENT-17

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

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

CODE:

```
print "Enter three numbers:\n";
chomp($a = <STDIN>);
chomp($b = <STDIN>);
chomp($c = <STDIN>);

$largest = $a;

if ($b > $largest) {
$largest = $b;
}

if ($c > $largest) {
$largest = $c;
}

print "The largest number is: $largest\n";
```

OUTPUT:

```
Enter three numbers:
5
9
3
The largest number is: 9
```

b).

```
sub print_table {
my ($num) = @_;
```

```
print "\nMultiplication Table of $num:\n";

for (my $i = 1; $i <= 10; $i++) {
print "$num x $i = " . ($num * $i) . "\n";
}
}

# Main program
for (my $n = 1; $n <= 10; $n++) {
print_table($n);
}
```

OUTPUT:

```
Multiplication Table of 1:
1 x 1 = 1
...
1 x 10 = 10

Multiplication Table of 2:
2 x 1 = 2
...
2 x 10 = 20
```

EXPERIMENT-18

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

a) Shift

b) Unshift

c)Push

CODE:

```
# Initialize a list
@numbers = (10, 20, 30);
print "Original List: @numbers\n";
# --- PUSH ---
# Adds elements to the end of the list
push(@numbers, 40, 50);
print "After push (40, 50): @numbers\n";
# --- UNSHIFT ---
# Adds elements to the beginning of the list
unshift(@numbers, 5);
print "After unshift (5): @numbers\n";
# --- SHIFT ---
# Removes element from the beginning of the list
$removed = shift(@numbers);
print "After shift (removed $removed): @numbers\n";
```

OUTPUT:

```
Original List: 10 20 30
After push (40, 50): 10 20 30 40 50
After unshift (5): 5 10 20 30 40 50
After shift (removed 5): 10 20 30 40 50
```

EXPERIMENT-19

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

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

CODE:

```
# Perl program to substitute a word in a string
print "Enter a string:\n";
chomp($string = <STDIN>);
print "Enter word to replace:\n";
chomp($old = <STDIN>);
print "Enter new word:\n";
chomp($new = <STDIN>);
# Substitute word
$string =~ s/\b$old\b/$new/g;
print "Updated string: $string\n";
```

OUTPUT:

```
Enter a string:
I love Perl programming
Enter word to replace:
Perl
Enter new word:
Python
Updated string: I love Python programming
```

b)

```
# Perl program to validate IP and Email
print "Enter an IP address:\n";
chomp($ip = <STDIN>);
print "Enter an Email address:\n";
chomp($email = <STDIN>);
# --- Validate IP Address ---
if ($ip =~ /^(\d{1,3}\.){3}\d{1,3}$/) {
    print "Valid IP address format\n";
} else {
    print "Invalid IP address format\n";
}
# --- Validate Email Address ---
if ($email =~ /^[\w\.-]+@[\w\.-]+\.\w+$/) {
    print "Valid Email address\n";
} else {
    print "Invalid Email address\n";
}
```

OUTPUT:

```
Enter an IP address:
192.168.1.1
Enter an Email address:
test@gmail.com
Valid IP address format
Valid Email addres
```

EXPERIMENT-20

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

CODE:

```
# Perl program to print file in reverse order using command line argument
# Check if filename is provided
if (@ARGV != 1) {
    die "Usage: perl reverse.pl <filename>\n";
}
$filename = $ARGV[0];
# Open file
open(my $fh, "<", $filename) or die "Cannot open file: $!";
# Read all lines into array
@lines = <$fh>;
# Close file
close($fh);
# Print lines in reverse order
@rev_lines = reverse(@lines);
print "File content in reverse order:\n";
print @rev_lines;
```

RUN: perl reverse.pl input.txt

INPUT:

Line 1
Line 2
Line 3

OUTPUT:

File content in reverse order:

Line 3

Line 2

Line 1