SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

PRACTICAL RECORD NOTE

STUDENT NAME :

REGISTER NUMBER :

CLASS : BCA SECTION: B

YEAR & SEMESTER : III YEAR & V SEMESTER

SUBJECT CODE : UCA23S05L

SUBJECT TITLE : LUA PROGRAMMING

OCTOBER 2025

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

FACULTY OF SCIENCE AND HUMANITIES

DEPARTMENT OF COMPUTER APPLICATIONS

SRM Nagar, Kattankulathur – 603 203

CERTIFICATE

Certified to be the bonafide record of practical work done by

_____

Register No._____of_____Degree course for

UCA23S05L – LUA PROGRAMMING in the Computer lab in SRM Institute of Science and Technology during the academic year 2025-2026.

Staff In-charge *Head of the Department*

Submitted for Semester Practical Examination held on_____.

Internal Examiner *External Examiner*

INDEX

EX NO: 01
DATE:

1.STRING MANIIPULATION

Aim :

To demonstrate various string manipulation operations in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file. Choose "Lua Script" as the file type.
3. Write a lua code to execute string manipulation.
4. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
5. In this program, various string functions such as converting strings to uppercase and lowercase, finding string length, reversing strings, extracting substrings, repeating strings, finding characters in a string, converting characters to their ASCII values and replacing specific substrings in a given string are demonstrated.
6. To execute the program, click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Code:

```
s1="helloabcdef"
s2="HELLO"
print (s1)
print ("upper", string.upper(s1))
print (string.lower (s2))
print (string.len (s1))
print (string.reverse (s1))
print (string.sub(s1,3))
print (string.sub(s1,3,5))
print (string.sub(s1,-3))
print(string.sub(s1,-3))
print (string.sub(s1,-3,-2))
print (string.rep(s2,5))
print (s1..s2)
print (string.find(s1, "def"))
print (string.byte("hai"))
print(string.byte(s1,1))
print (string.char(104))
print (string.gsub (s1, "abc", "xyz"))
```

OUTPUT:

```
helloabcdef
upper   HELLOABCDEF
hello
11
fedcbaolleh
lloabcdef
llo
def
def
de
HELLOHELLOHELLOHELLOHELLO
helloabcdefHELLO
9       11
104
104
h
```

Result:

The String Manipulation operations is executed successfully.

EX NO: 02 **TO FIND AVERAGE USING FUNCTION**
DATE:

Aim :

To find average of numbers using function in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file. Choose "Lua Script" as the file type.
3. Write a lua code to execute to find average using function .
4. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
5. The output obtained is the average of the given numbers.
6. To execute the program , click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

CODE:

```
function average(...)
result = 0 local arg =
{...} for i,v in
ipairs(arg) do result =
result + v  end
return result/#arg
end
print ("The average is", average(10,3,5,4,5,6))
```

OUTPUT:

The average is 5.5

Result:

The program completed successfully.

EX NO: 03
DATE:

ARITHMETIC OPERATIONS

Aim :

To perform various arithmetic operations between two numbers in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file. Choose "Lua Script" as the file type.
3. Write a lua code to execute arithmetic operations.
4. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
5. In this program, we perform various arithmetic operations like +,-,*,/,%,^ on the given output and print the obtained result.
6. To execute the program, click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

```
a = 20
b = 10
c = a + b

print("Line 1 - Value of c is ", c )
c = a - b
print("Line 2 - Value of c is ", c )
c = a * b
print("Line 3 - Value of c is ", c )
c = a / b
print("Line 4 - Value of c is ", c )
c = a % b
print("Line 5 - Value of c is ", c )
c = a^2
print("Line 6 - Value of c is ", c )
c = -a
print("Line 7 - Value of c is ", c )
```

OUTPUT:

```
Line 1 - value of c is  30
Line 2 - value of c is  10
Line 3 - value of c is  200
Line 4 - value of c is  2.0
Line 5 - value of c is  0
Line 6 - value of c is  400.0

[Execution complete with exit code 0]
```

Result:

The arithmetic Operations program executed successfully.

EX NO: 04 **CALCULATION OF GRADE USING ELSEIF LADDER**
DATE:

Aim:

To calculate the grade based on the percentage using elseif ladder in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file. Choose "Lua Script" as the file type.
3. Write a lua code to execute else-if ladder.
4. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
5. In this program, grade is generated based on the percentage using elseif ladder.
6. To execute the program , click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

```
local percentage = 65
if percentage >= 90 then
   print("Grade: A")
elseif percentage >= 80 then
   print("Grade: B")
elseif percentage >= 70 then
   print("Grade: C")
elseif percentage >= 60 then
   print("Grade: D")
else
   print("Grade: F")
end
```

Output:

Your Grade is D

Result:

The program is successfully completed.

EX NO: 05
DATE:

STRING REVERSE

Aim :

To print the reverse of the given string in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file.
3. Choose "Lua Script" as the file type.
4. Write a lua code to execute string reverse .
5. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
6. In this program, the number is reversed using number.reverse function and output is generated.
7. To execute the program , click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

```
number = "123456"
print(string.find(number,"123456"))
reversedNumber = number.reverse(number)
print("The new number is ",reversedNumber)
```

Output:

```
1       6
The new number is   654321
```

Result:

The string reverse function program is executed successfully.

EX NO: 06 **COPYING THE CONTENT OF ONE TABLE TO ANOTHER TABLE**
DATE:

Aim:

To copy the content of table to another table based on the given data in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file. Choose "Lua Script" as the file type.
3. Write a lua code to execute tables in lua.
4. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
5. In this program, table is generated based on the given data.
6. To execute the program , click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

```
mytable = {}
 print("Type of mytable is", type(mytable))
mytable[1] = "Lua"
mytable["wow"] = "Tutorial"
print("mytable Element at index 1 is", mytable[1])
print("mytable Element at index wow is", mytable["wow"])
alternatable = mytable
print("alternatable Element at index 1 is ", alternatable[1])
print("alternatable Element at index wow is ", alternatable["wow"])
alternatable["wow"] = "I changed it"
print("mytable Element at index wow is", mytable["wow"])
alternatable = nil
print("alternatable is ", alternatable)
print("mytable Element at index wow is", mytable["wow"])
mytable = nil
print("mytable is", mytable)
```

OUTPUT:

```
Type of mytable is      table
mytable Element at index 1 is   Lua
mytable Element at index wow is Tutorial
alternatable Element at index 1 is      Lua
alternatable Element at index wow is    Tutorial
mytable Element at index wow is I changed it
alternatable is         nil
mytable Element at index wow is I changed it
mytable is      nil
```

Result:

The table program is executed successfully**.**

EX NO: 07
DATE:

STRING CONCATENATION

Aim :

To perform operations like concatenation, insertion and sorting on given string in Table.

Procedure:

1. Open LuaEdit 2010.

2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file. Choose "Lua Script" as the file type.

3. Write a lua code to execute string concatenation.

4. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.

5. In this program, operations like concatenation, insertion and sorting are performed on the string and the output is printed.

6. To execute the program, click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

PROGRAM:

```
fruits = {"banana", "orange", "apple"} print("Concatenated String",
table.concat(fruits)) print("Concatenated String",
table.concat(fruits,",")) print("Concatenated String",
table.concat(fruits,",",2,3))
-- [Insertion]
fruits = {"banana","orange","apple"}
table.insert(fruits,"mango")
print("Fruit at index 4 is ",fruits[4])
table.insert(fruits,2,"grapes")
print("Fruit at index 2 is ",fruits[2])
print("The maximum elements in table is",table.maxn(fruits))
print("The last element is",fruits[5])
table.remove(fruits)
print("The previous last element is",fruits[5])
--[Sorting]
fruits = {"banana","orange","apple","grapes"}
for k,v in ipairs(fruits) do
print(k,v)
end
table.sort(fruits)
print("sorted table")
for k,v in ipairs(fruits) do
print(k,v)
end
```

OUTPUT:

```
Concatenated String     bananaorangeapple
Concatenated String     banana,orange,apple
Concatenated String     orange,apple

[Execution complete with exit code 0]
```

```
Fruit at index 4 is     mango
Fruit at index 2 is     grapes
The maximum elements in table is        5
The last element is     mango
The previous last element is    nil
```

```
1       banana
2       orange
3       apple
4       grapes
sorted table
1       apple
2       banana
3       grapes
4       orange
```

Result:

The Table manipulation program executed successfully.

EX NO: 08 **MATRIX ADDITION USING ARRAY**

DATE:

Aim :

To Add the two matrix using array in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file. Choose "Lua Script" as the file type.
3. Write a lua code to execute arrays in lua.
4. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
5. In this program, concept of matrix addition is implemented.
6. To execute the program , click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

```
function matrixAddition(matrix1, matrix2)
   local result = {}
      local rows = #matrix1
   local cols = #matrix1[1]
   if rows ~= #matrix2 or cols ~= #matrix2[1] then
      return nil
   end
      for i = 1, rows do
      result[i] = {}
      for j = 1, cols do
         result[i][j] = matrix1[i][j] + matrix2[i][j]
      end
   end
     return result
end
local matrix1 = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}
local matrix2 = {{9, 8, 7}, {6, 5, 4}, {3, 2, 1}}
local result = matrixAddition(matrix1, matrix2)
```

```
if result then
    print("Matrix 1:")
    for _, row in ipairs(matrix1) do
        print(table.concat(row, "\t"))
    end
    print("\nMatrix 2:")
    for _, row in ipairs(matrix2) do
        print(table.concat(row, "\t"))
    end
   print("\nResultant Matrix (Matrix 1 + Matrix 2):")
    for _, row in ipairs(result) do
        print(table.concat(row, "\t"))
    end
else
    print("Matrices cannot be added due to different dimensions.")
end
```

OUTPUT:

```
Matrix 1:
1    2    3
4    5    6
7    8    9

Matrix 2:
9    8    7
6    5    4
3    2    1

Resultant Matrix (Matrix 1 + Matrix 2):
10   10   10
10   10   10
10   10   10
```

Result:

The Matrix Addition program is executed successfully.

EX NO: 09
DATE:

ITERATORS

Aim :

To demonstrate the use of iterators in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file.
3. Choose "Lua Script" as the file type.
4. Write a lua code to execute.
5. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
6. To execute the program, click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

[StateFullIterators]

```
array = {"Lua", "Tutorial"}
 function elementIterator (collection)
local index = 0
 local count =#collection
 return function ()
 index = index + 1
if index <= count
then
 return collection[index]
end
 end
 end
 for element in elementIterator(array)
 do
print(element)
 end
```

[StateLessIterators]

```
function square(iteratorMaxCount,currentNumber)
if currentNumber<iteratorMaxCount
then currentNumber = currentNumber+1
 return currentNumber, currentNumber*currentNumber
 end
end
 for i,n in square,3,0
do
 print(i,n)
 end
```

OUTPUT:

```
Lua
Tutorial

[Execution complete with exit code 0]
```

```
1       1
2       4
3       9

[Execution complete with exit code 0]
```

Result:

The Iterator program is completed successfully.

EX NO: 10 **FIND AREA OF SQUARE AND RECTANGLE**
DATE:

Aim :

To find area of square and area of rectangle in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file.
3. Choose "Lua Script" as the file type.
4. Write a lua code to execute .
5. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
6. To execute the program , click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

```
Shape = {area = 0}
function Shape:new (o,side)
o = o or {}
setmetatable(o,self)
self._index = self
side = side or 0
self.area = side*side;
return o
end
function Shape:printArea ()
print("The area is ", self.area)
end
myshape = Shape:new(nil,10)
myshape:printArea()
Square = Shape:new()
function Square:new (o,side)
o = o or Shape:new(o, side)
setmetatable(o,       self)
self._index = self
return o
end
```

```
 function Square:printArea ()
print("The area of square is", self.area)
end
mysquare = Square:new(nil, 10)
mysquare:printArea()
Rectangle = Shape:new()
function Rectangle:new (o, length,breadth)
 o = o or Shape:new(o)
setmetatable(o, self)
self._index = self
self.area = length * breadth
 return o
 end
function Rectangle:printArea ()
print("The area of Rectangle is ", self.area)
end
myrectangle = Rectangle:new(nil,10,20)
myrectangle:printArea()
```

OUTPUT:

```
The area is      100
The area of square is    100
The area of Rectangle is         200

[Execution complete with exit code 0]
```

Result:

The program is completed successfully.

EX NO: 11 **TO CHECK STUDENT RESULT USING IF ELSE STATEMENT**
DATE:

Aim :

To check student result using if else statement in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file.
3. Choose "Lua Script" as the file type.
4. Write a lua code to execute .
5. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
6. To execute the program , click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

```
m1=36
m2=45
m3=56
m4=56
if m1>=35 and m2>=35 and m3>=35 and m4>=35 then
total=m1+m2+m3+m4
avg=total/4
print("total",total)
print("avg",avg)
else
print"fail"
end
```

Output:

Total= 193

Avg=48.25

Result:

The program is completed successfully.

EX NO: 12 **TO CHECK THE CHARACTER IS VOWEL OR NOT**
DATE:

Aim :

To check whether the character is vowel or not in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file.
3. Choose "Lua Script" as the file type.
4. Write a program to check the character is vowel code to execute .
5. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
6. To execute the program, click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

PROGRAM:

```
local ra = "a"
if ra == 'a' or ra == 'e' or ra == 'i' or ra == 'o' or ra == 'u' or ra == 'A' or ra == 'E' or
 ra == 'I' or  ra == 'O' or ra == 'U' then
 print("vowel")
else
print("not vowel")
end
```

Output:

1. a=”a”

   vowel

2. a=”t”

   not vowel

Result:

The vowels program is executed successfully.

EX NO: 13

BUBBLE SORT

DATE:

Aim :

To write a bubble sort program in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file.
3. Choose "Lua Script" as the file type.
4. Write a bubble sort code to execute .
5. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
6. To execute the program, click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program

```
function bubbleSort(arr)
local n = #arr
 local swapped
repeat
 swapped = false
for i = 1, n - 1 do
 if arr[i] > arr[i + 1] then
 arr[i], arr[i + 1] = arr[i + 1], arr[i]
  swapped = true
  end
 end
  n = n - 1
 until not swapped
end
local numbers = {64, 34, 25, 12, 22, 11, 90}
print("Original array: ", table.concat(numbers, ", "))
bubbleSort(numbers)
print("Sorted array using Bubble Sort: ", table.concat(numbers, ", "))
```

Output:

Original array: 64, 34, 25, 12, 22, 11, 90

Sorted array using Bubble Sort:11, 12, 22, 25, 34, 64, 90

Result:

The program is completed successfully.

EX NO: 14 **TO GENERATE FIBONACCI SERIES USING LOOP**

DATE:

Aim :

To write a Fibonacci series program using loop in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file.
3. Choose "Lua Script" as the file type.
4. Write a Fibonacci series code to execute .
5. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
6. To execute the program, click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program

```
function fibonacciLoop(n)
local fib = {0, 1}
for i = 3, n do
fib[i] = fib[i - 1] + fib[i - 2]
end
return fib
end
local n = 10
local fibSeries = fibonacciLoop(n)
print("Fibonacci Series using Loop:")
print(table.concat(fibSeries, ", "))
```

Output:

Fibonacci Series using Loop: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34

Result:

The Fibonacci Series program is completed successfully

EX NO: 15 **TO PRINT STUDENT INFORMATION USING INHERITANCE**
DATE:
Aim :

To print student information using inheritance concept in LUA.

Procedure:

1. Open LuaEdit 2010.
2. To create a new file, Click on "File" in the menu bar. Select "New" to create a new file.
3. Choose "Lua Script" as the file type.
4. Print a student information using inheritance code to execute .
5. Click on "File" in the menu bar. Select "Save" or "Save As" to save your Lua script with a .lua extension.
6. To execute the program, click on "Tools" in the menu bar and select "Run Script". You can view the output in the output console.

Program:

```
local Person = {
name = "",
age = 0,
}
function Person:new(name, age)
newObj = {}
self.__index = self
setmetatable(newObj, self)
newObj.name = name
newObj.age = age
return newObj
end
function Person:displayInfo()
print("Name:", self.name)
print("Age:", self.age)
end
local Student = Person:new()
```

```
function Student:new(name, age, grade)
newObj = Person:new(name, age)
self.__index = self
setmetatable(newObj, self)
newObj.grade = grade
return newObj
end
function Student:displayInfo()
print("Name:", self.name)
print("Age:", self.age)
print("Grade:",  self.grade)
end
local person = Person:new("John Doe", 30,”B”)
local student = Student:new("Alice Smith", 25, "A")
print("student Information:")
person:displayInfo()
print("\nStudent Information:")
student:displayInfo()
```

Output :

student Information:
Name: John Doe
Age: 30
Grade:B

Student Information:
Name: Alice Smith
Age: 25
Grade: A

Result:

The inheritance program is executed successfully.