Assignment-1

1. WAP to add two numbers.

```
# Input
print "Enter first number: "
num1 = gets.chomp.to_i

print "Enter second number: "
num2 = gets.chomp.to_i

# Addition
sum = num1 + num2
puts "The sum of #{num1} and #{num2} is: #{sum}"
```

output:

```
Enter first number: 12
Enter second number: 34
The sum of 12 and 34 is: 46
```

2. WAP to take name as input and print “Hello <name>”

```
print "Enter your name: "
name = gets.chomp
puts "Hello #{name}!"
```

output:

```
Enter your name: John
Hello John!
```

3. WAP to find the area of trapezoid, area=(b1+b2)/2*h

```
print "Enter the length of the first base (b1): "
b1 = gets.chomp.to_f

print "Enter the length of the second base (b2): "
```

```
b2 = gets.chomp.to_f

print "Enter the height (h): "
h = gets.chomp.to_f

# Calculate area
area = (b1 + b2) / 2 * h
puts "The area of the trapezoid is: #{area}"
```

output:

```
Enter the length of the first base (b1): 5
Enter the length of the second base (b2): 7
Enter the height (h): 4
The area of the trapezoid is: 24.0
```

4. WAP to calculate Simple Interest, si=(p*t*r)/100

```
print "Enter the principal amount (P): "
principal = gets.chomp.to_f

print "Enter the time period in years (T): "
time = gets.chomp.to_f

print "Enter the rate of interest per annum (R): "
rate = gets.chomp.to_f

# Calculate Simple Interest
simple_interest = (principal * time * rate) / 100
puts "The Simple Interest is: #{simple_interest}"
```

output:

```
Enter the principal amount (P): 1000
```

Enter the time period in years (T): 2

Enter the rate of interest per annum (R): 5

The Simple Interest is: 100.0

5. WAP to find m to the power n.

```
print "Enter the base (m): "
m = gets.chomp.to_i

print "Enter the exponent (n): "
n = gets.chomp.to_i

# Calculate m raised to the power of n
result = m ** n
puts "#{m} raised to the power of #{n} is: #{result}"
```

output:

Enter the base (m): 2

Enter the exponent (n): 3

2 raised to the power of 3 is: 8

6. WAP to find area of circle. [Hint: Use pi function]

```
# Define the value of pi
PI = 3.14159

# Input
print "Enter the radius of the circle: "
radius = gets.chomp.to_f

# Calculate the area of the circle
area = PI * radius**2
puts "The area of the circle with radius #{radius} is: #{area}"
```

output:

Enter the radius of the circle: 5

The area of the circle with radius 5.0 is: 78.53975

7. WAP to find perimeter of a Rectangle.

```
print "Enter the length of the rectangle: "
length = gets.chomp.to_f

print "Enter the width of the rectangle: "
width = gets.chomp.to_f

# Calculate the perimeter of the rectangle
perimeter = 2 * (length + width)
puts "The perimeter of the rectangle is: #{perimeter}"
```

output:

Enter the length of the rectangle: 6

Enter the width of the rectangle: 4

The perimeter of the rectangle is: 20.0

8. WAP to find the Simple Interest and the total amount when the Principal, Rate of Interest and Time are entered by the user.

```
print "Enter the principal amount (P): "
principal = gets.chomp.to_f

print "Enter the rate of interest per annum (R): "
rate = gets.chomp.to_f

print "Enter the time period in years (T): "
time = gets.chomp.to_f

# Calculate Simple Interest
```

```
simple_interest = (principal * rate * time) / 100

# Calculate Total Amount
total_amount = principal + simple_interest
puts "Simple Interest: #{simple_interest}"
puts "Total Amount: #{total_amount}"
```

output:

```
Enter the principal amount (P): 1000
Enter the rate of interest per annum (R): 5
Enter the time period in years (T): 2
Simple Interest: 100.0
Total Amount: 1100.0
```

9. WAP to convert °C to °F and °F to °C.

```
# Function to convert Celsius to Fahrenheit
def celsius_to_fahrenheit(celsius)
  return (celsius * 9 / 5) + 32
end

# Function to convert Fahrenheit to Celsius
def fahrenheit_to_celsius(fahrenheit)
  return (fahrenheit - 32) * 5 / 9
end
print "Enter temperature in Celsius (°C): "
celsius = gets.chomp.to_f

print "Enter temperature in Fahrenheit (°F): "
fahrenheit = gets.chomp.to_f

# Convert Celsius to Fahrenheit
converted_fahrenheit = celsius_to_fahrenheit(celsius)
```

```
# Convert Fahrenheit to Celsius
converted_celsius = fahrenheit_to_celsius(fahrenheit)
puts "#{celsius}°C is equal to #{converted_fahrenheit}°F"
puts "#{fahrenheit}°F is equal to #{converted_celsius}°C"
```

output:

```
Enter temperature in Celsius (°C): 25
Enter temperature in Fahrenheit (°F): 77
25.0°C is equal to 77.0°F
77.0°F is equal to 25.0°C
```

10. WAP to find distance between two points.

```
print "Enter the x-coordinate of the first point: "
x1 = gets.chomp.to_f

print "Enter the y-coordinate of the first point: "
y1 = gets.chomp.to_f

print "Enter the x-coordinate of the second point: "
x2 = gets.chomp.to_f

print "Enter the y-coordinate of the second point: "
y2 = gets.chomp.to_f

# Calculate distance between the two points
distance = Math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
puts "The distance between the two points is: #{distance}"
```

output:

```
Enter the x-coordinate of the first point: 3
Enter the y-coordinate of the first point: 4
Enter the x-coordinate of the second point: 7
```

Enter the y-coordinate of the second point: 1

The distance between the two points is: 5.0

11. WAP for Pythagoras Theorem.

```
print "Enter the length of side 'a': "
a = gets.chomp.to_f

print "Enter the length of side 'b': "
b = gets.chomp.to_f

# Calculate the length of the hypotenuse (c)
c = Math.sqrt(a**2 + b**2)
puts "The length of the hypotenuse (c) is: #{c}"
```

output:

Enter the length of side 'a': 3

Enter the length of side 'b': 4

The length of the hypotenuse (c) is: 5.0

12. If you run a 10 kilometer race in 43 minutes 30 seconds, what is your average time per mile? What is your average speed in miles per hour? (Hint: there are 1.61 kilometers in a mile).

```
# Constants
total_distance_km = 10
total_time_minutes = 43 * 60 + 30  # Convert minutes to seconds

# Convert kilometers to miles
total_distance_miles = total_distance_km / 1.61

# Calculate average time per mile in seconds
average_time_per_mile_seconds = total_time_minutes / total_distance_miles
```

```
# Convert seconds to minutes and seconds
average_time_per_mile_minutes = average_time_per_mile_seconds / 60
average_time_per_mile_seconds %= 60

# Calculate average speed in miles per hour
average_speed_mph = total_distance_miles / (total_time_minutes / 60)
puts "Average time per mile: #{average_time_per_mile_minutes.to_i} minutes #{average_time_per_mile_seconds.to_i} seconds"
puts "Average speed: #{'%.2f' % average_speed_mph} miles per hour"
```

output:

Average time per mile: 7 minutes 2 seconds

Average speed: 8.57 miles per hour

13. Suppose the cover price of a book is $24.95, but bookstores get a 40% discount. Shipping costs $3 for the first copy and 75 cents for each additional copy. What is the total wholesale cost for 60 copies?

```
# Constants
cover_price = 24.95
discount = 0.4
shipping_first_copy = 3
shipping_additional_copy = 0.75
total_copies = 60

# Calculate discounted price per copy
discounted_price = cover_price * (1 - discount)

# Calculate total wholesale cost
total_wholesale_cost = (discounted_price * total_copies) + shipping_first_copy + (total_copies - 1) * shipping_additional_copy
```

```
puts "Total wholesale cost for #{total_copies} copies: $#{'%.2f' % total_wholesale_cost}"
```

output:

Total wholesale cost for 60 copies: $945.45

14. If I leave my house at 6:52 am and run 1 mile at an easy pace (8:15 per mile), then 3 miles at tempo (7:12 per mile) and 1 mile at easy pace again, what time do I get home for breakfast?

```
# Define start time
start_time = Time.new(2024, 3, 8, 6, 52, 0) # 6:52 am

# Define paces in seconds per mile
easy_pace = 8 * 60 + 15 # 8 minutes 15 seconds per mile
tempo_pace = 7 * 60 + 12 # 7 minutes 12 seconds per mile

# Calculate total time taken for the run
total_time_seconds = (easy_pace * 2 + tempo_pace * 3) # Total time in seconds

# Calculate time to return home
return_time = start_time + total_time_seconds
puts "Time to return home for breakfast: #{return_time.strftime('%I:%M %p')}"
```

output:

Time to return home for breakfast: 07:30 AM

Assignment-2

1. WAP to find greatest of three numbers

```
# Input
print "Enter the first number: "
num1 = gets.chomp.to_i
```

```
print "Enter the second number: "
num2 = gets.chomp.to_i

print "Enter the third number: "
num3 = gets.chomp.to_i

# Find the greatest number
greatest = num1
if num2 > greatest
  greatest = num2
end
if num3 > greatest
  greatest = num3
end

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

Output:

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

2. WAP to find whether a given number is even or not.

```
# Input
print "Enter a number: "
number = gets.chomp.to_i

# Check if the number is even
if number % 2 == 0
  puts "#{number} is an even number."
```

```
else
  puts "#{number} is not an even number."
end
```

output:

```
Enter a number: 10
10 is an even number.
```

3. WAP to whether a given number is prime or not.

```
# Function to check if a number is prime
def is_prime(number)
  if number <= 1
    return false
  elsif number <= 3
    return true
  elsif number % 2 == 0 || number % 3 == 0
    return false
  else
    i = 5
    while i * i <= number
      if number % i == 0 || number % (i + 2) == 0
        return false
      end
      i += 6
    end
    return true
  end
end

# Input
print "Enter a number: "
num = gets.chomp.to_i
```

```
# Check if the number is prime
if is_prime(num)
  puts "#{num} is a prime number."
else
  puts "#{num} is not a prime number."
end
```

output:

```
Enter a number: 7
7 is a prime number.
```

4. WAP to find whether a given string or number is

palindrome or not

```
# Function to check if a string is palindrome
def is_palindrome(input)
  # Convert input to string and remove non-alphanumeric characters
  cleaned_input = input.to_s.downcase.gsub(/\W/, '')

  # Check if the cleaned input is equal to its reverse
  return cleaned_input == cleaned_input.reverse
end

# Input
print "Enter a string or number: "
input = gets.chomp

# Check if the input is palindrome
if is_palindrome(input)
  puts "#{input} is a palindrome."
else
  puts "#{input} is not a palindrome."
```

```
end
```

output:

```
Enter a string or number: radar
radar is a palindrome.
```

5. Given 3 sides, WAP to find whether a given triangle is right angled or not.

```
# Function to check if a triangle is right-angled
def is_right_triangle(a, b, c)
  sides = [a, b, c].sort
  return sides[0]**2 + sides[1]**2 == sides[2]**2
end

# Input
print "Enter the length of side 'a': "
a = gets.chomp.to_f

print "Enter the length of side 'b': "
b = gets.chomp.to_f

print "Enter the length of side 'c': "
c = gets.chomp.to_f

# Check if the triangle is right-angled
if is_right_triangle(a, b, c)
  puts "The triangle with sides #{a}, #{b}, and #{c} is a right-angled triangle."
else
  puts "The triangle with sides #{a}, #{b}, and #{c} is not a right-angled triangle."
end
```

output:

```
Enter the length of side 'a': 4
```

Enter the length of side 'b': 5

Enter the length of side 'c': 6

The triangle with sides 4.0, 5.0, and 6.0 is not a right-angled triangle.

6. WAP to print even from 1 through 100

```
# Print even numbers from 1 through 100
(1..100).each do |num|
  if num.even?
    puts num
  end
end
```

output:

2

4

6

8

10

12

14

16

18

20

22

24

26

28

30

32

34

36

38

40

42

44

46

48

50

52

54

56

58

60

62

64

66

68

70

72

74

76

78

80

82

84

86

88

90

92

94

96

98

100

7. WAP to print odd from 1 through 100

```
# Print odd numbers from 1 through 100
(1..100).each do |num|
  if num.odd?
    puts num
  end
end
```

output:

1

3

5

7

9

11

13

15

17

19

21

23

25

27

29

31

33

35

37

39

41

43

45

47

49

51

53

55

57

59

61

63

65

67

69

71

73

75

77

79

81

83

85

87

89

91

93

95

97

99

8. WAP to print prime numbers 1 through 100.

Function to check if a number is prime

```
def is_prime(num)
  return false if num <= 1
  return true if num <= 3
  return false if num % 2 == 0 || num % 3 == 0
  i = 5
  while i * i <= num
    return false if num % i == 0 || num % (i + 2) == 0
    i += 6
  end
  return true
end

# Print prime numbers from 1 through 100
(1..100).each do |num|
  if is_prime(num)
    puts num
  end
end
```

output:

```
2
3
5
7
11
13
17
19
23
29
31
37
```

41

43

47

53

59

61

67

71

73

79

83

89

97

9. WAP to print table of 7

```
# Print table of 7
puts "Table of 7:"
(1..10).each do |i|
  puts "7 x #{i} = #{7 * i}"
end
```

output:

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

10. WAP to print reverse table of 7

```
# Print reverse table of 7
puts "Reverse Table of 7:"
(1..10).reverse_each do |i|
  puts "7 x #{i} = #{7 * i}"
end
```

Output:

Reverse Table of 7:

7 x 10 = 70

7 x 9 = 63

7 x 8 = 56

7 x 7 = 49

7 x 6 = 42

7 x 5 = 35

7 x 4 = 28

7 x 3 = 21

7 x 2 = 14

7 x 1 = 7

11. WAP to print table of 1 to 10 in a tabular format.

```
# Print tables of numbers from 1 to 10 in a tabular format
puts "Tables from 1 to 10:"

(1..10).each do |i|
  table_row = ""
  (1..10).each do |j|
    table_row += "#{i} x #{j} = #{i * j}\t"
  end
  puts table_row
```

end

output:

Tables from 1 to 10:

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

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

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

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

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

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

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

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

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

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

12. WAP to print all prime numbers in an Interval.

```
# Function to check if a number is prime
def is_prime(num)
  return false if num <= 1
  return true if num <= 3
  return false if num % 2 == 0 || num % 3 == 0
  i = 5
  while i * i <= num
    return false if num % i == 0 || num % (i + 2) == 0
    i += 6
```

```
  end
  return true
end

# Input
print "Enter the lower bound of the interval: "
lower_bound = gets.chomp.to_i

print "Enter the upper bound of the interval: "
upper_bound = gets.chomp.to_i

# Print prime numbers within the interval
puts "Prime numbers within the interval [#{lower_bound}, #{upper_bound}]:"
(lower_bound..upper_bound).each do |num|
  if is_prime(num)
    puts num
  end
end
```

output:

```
Enter the lower bound of the interval: 10
Enter the upper bound of the interval: 50
Prime numbers within the interval [10, 50]:
11
13
17
19
23
29
31
37
41
```

43

47

13. WAP to guess a number between 1 and 9. User is prompted to enter a guess. If the user guesses wrong then the prompt appears again until the guess is correct, on successful guess, user will get a "Well guessed!" message, and the program will exit

```
# Generate a random number between 1 and 9
target_number = rand(1..9)

# Prompt the user to guess the number
puts "Guess a number between 1 and 9:"

loop do
  print "Enter your guess: "
  guess = gets.chomp.to_i

  if guess == target_number
    puts "Well guessed!"
    break
  else
    puts "Try again! Wrong guess."
  end
end
```

output:

```
Guess a number between 1 and 9:
Enter your guess: 5
Try again! Wrong guess.
Enter your guess: 3
Try again! Wrong guess.
```

Enter your guess: 7

Well guessed!

Assignment-3

WAP for the following:

1. make_bricks

We want to make a row of bricks that is goal inches long.

We have a number of small bricks (1 inch each) and big

bricks (5 inches each). WAF make_bricks(small, big,

goal) that returns True if it is possible to make the goal

by choosing from the given bricks.

make_bricks(3, 1, 8) → True

make_bricks(3, 1, 9) → False

make_bricks(3, 2, 10) → True

```
def make_bricks(small, big, goal)
  max_big_needed = goal / 5
  big_used = [max_big_needed, big].min
  remaining_goal = goal - (big_used * 5)

  return small >= remaining_goal
end

# Test cases
puts make_bricks(3, 1, 8)
puts make_bricks(3, 1, 9)
puts make_bricks(3, 2, 10)
```

output:

true

false

true

2. lone_sum

Given 3 int values, a b c, WAF lone_sum(a, b, c) that returns their sum. However, if one of the values is the same as another of the values, it does not count towards the sum.

lone_sum(1, 2, 3) → 6

lone_sum(3, 2, 3) → 2

lone_sum(3, 3, 3) → 0

```
def lone_sum(a, b, c)
  sum = 0

  if a != b && a != c
    sum += a
  end

  if b != a && b != c
    sum += b
  end

  if c != a && c != b
    sum += c
  end

  return sum
end

# Test cases
puts lone_sum(1, 2, 3)
puts lone_sum(3, 2, 3)
puts lone_sum(3, 3, 3)
```

output:

6

2

0

1. WAP that will keep track of items for a shopping list.
The program should keep asking for new items until
nothing is entered (no input followed by enter/return
key). The program should then display the full shopping
list.

```
# Initialize an empty array to store the shopping list items
shopping_list = []

# Keep asking for new items until nothing is entered
loop do
  print "Enter item for the shopping list (or press Enter/Return to finish): "
  item = gets.chomp

  break if item.empty? # Exit the loop if nothing is entered

  # Add the item to the shopping list
  shopping_list << item
end

# Display the full shopping list
puts "\nShopping List:"
if shopping_list.empty?
  puts "No items in the shopping list."
else
  shopping_list.each_with_index do |item, index|
    puts "#{index + 1}. #{item}"
```

```
  end
end
```

output:

```
Enter item for the shopping list (or press Enter/Return to finish): Apples
Enter item for the shopping list (or press Enter/Return to finish): Milk
Enter item for the shopping list (or press Enter/Return to finish): Bread
Enter item for the shopping list (or press Enter/Return to finish):
Shopping List:
1. Apples
2. Milk
3. Bread
```

2. WAP that will store the schedule for a given day for a particular TV station. The program should ask you for the name of the station and the day of the week before asking you for the name of each show and the start and stop times. Once the schedule is complete it should be displayed as a table.

```
# Initialize a hash to store the schedule
schedule = {}

# Prompt for the name of the TV station and the day of the week
print "Enter the name of the TV station: "
station_name = gets.chomp

print "Enter the day of the week: "
day_of_week = gets.chomp

# Keep asking for the name of each show and the start and stop times
loop do
  print "Enter the name of the show (or press Enter/Return to finish): "
```

```
  show_name = gets.chomp

  break if show_name.empty? # Exit the loop if nothing is entered

  print "Enter the start time (hh:mm): "
  start_time = gets.chomp

  print "Enter the stop time (hh:mm): "
  stop_time = gets.chomp

  # Add the show to the schedule
  schedule[show_name] = { start_time: start_time, stop_time: stop_time }
end

# Display the schedule as a table
puts "\n#{station_name} Schedule for #{day_of_week}:"
puts "----------------------------------------"
puts "| Show Name\t| Start Time\t| Stop Time\t|"
puts "----------------------------------------"
schedule.each do |show_name, times|
  puts "| #{show_name}\t| #{times[:start_time]}\t| #{times[:stop_time]}\t|"
end
puts "----------------------------------------"
```

output:

```
Enter the name of the TV station: Channel X
Enter the day of the week: Monday
Enter the name of the show (or press Enter/Return to finish): News Hour
Enter the start time (hh:mm): 18:00
Enter the stop time (hh:mm): 19:00
Enter the name of the show (or press Enter/Return to finish): Movie Night
Enter the start time (hh:mm): 19:00
```

Enter the stop time (hh:mm): 22:00

Enter the name of the show (or press Enter/Return to finish):

Channel X Schedule for Monday:

```
-----------------------------------------
| Show Name     | Start Time   | Stop Time   |
-----------------------------------------
| News Hour     | 18:00        | 19:00       |
| Movie Night   | 19:00        | 22:00       |
-----------------------------------------
```

3. WAP to multiply two matrix supplied by user in rowmajor representation.

```
# Function to prompt user for matrix input
def get_matrix(rows, cols)
  matrix = []
  puts "Enter the elements of the matrix (#{rows}x#{cols}):"
  rows.times do |i|
    print "Row #{i + 1}: "
    row = gets.chomp.split.map(&:to_i)
    if row.length != cols
      puts "Error: Number of elements in each row should be #{cols}. Please try again."
      redo
    end
    matrix << row
  end
  return matrix
end

# Function to multiply two matrices
def multiply_matrices(matrix1, matrix2)
  result = []
```

```
  matrix1.each do |row1|
    temp_row = []
    matrix2.transpose.each do |col2|
      temp_row << row1.zip(col2).map { |a, b| a * b }.sum
    end
    result << temp_row
  end
  return result
end

# Prompt user for the dimensions of the first matrix
print "Enter the number of rows of the first matrix: "
rows1 = gets.chomp.to_i
print "Enter the number of columns of the first matrix: "
cols1 = gets.chomp.to_i

# Prompt user for the dimensions of the second matrix
print "Enter the number of rows of the second matrix: "
rows2 = gets.chomp.to_i
print "Enter the number of columns of the second matrix: "
cols2 = gets.chomp.to_i

# Check if multiplication is possible
if cols1 != rows2
  puts "Error: The number of columns of the first matrix must be equal to the number of rows of the second matrix for multiplication."
else
  # Prompt user for the elements of the first matrix
  matrix1 = get_matrix(rows1, cols1)

  # Prompt user for the elements of the second matrix
```

```
  matrix2 = get_matrix(rows2, cols2)

  # Multiply the matrices
  result_matrix = multiply_matrices(matrix1, matrix2)

  # Display the result
  puts "Result of matrix multiplication:"
  result_matrix.each { |row| puts row.join(' ') }
end
```

output:

```
Enter the number of rows of the first matrix: 2
Enter the number of columns of the first matrix: 2
Enter the number of rows of the second matrix: 2
Enter the number of columns of the second matrix: 2
Enter the elements of the matrix (2x2):
Row 1: 1 2
Row 2: 3 4
Enter the elements of the matrix (2x2):
Row 1: 5 6
Row 2: 7 8
Result of matrix multiplication:
19 22
43 50
```