1a. Develop a Julia program to simulate a calculator (for integer and real numbers).

```
function calcuator()
    println("Enter a valid arithmetic expression")
    o1,op,o2=split(readline())
    o1=parse(Float64,o1)
    o2=parse(Float64,o2)
    if op=="+"
        return o1+o2
    elseif op=="-"
        return o1-o2
    elseif op=="*"
        return o1*o2
    elseif op=="/"
        return o1/o2
    else
        println("Invalid operator")
    end
end
println(calcuator())
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter a valid arithmetic expression
3 + 5
8.0
Enter a valid arithmetic expression
4 - 5
-1.0
Enter a valid arithmetic expression
7 * 9
63.0
Enter a valid arithmetic expression
8 / 3
2.6666666666666665
```

```
Enter a valid arithmetic expression
2 $ 7
Invalid operator
nothing
```

1b. Develop a Julia program to add, subtract, multiply and divide complex numbers.

```
function complexarithmetic()
    println("Enter first complex number...")
    z1r,z1i=[parse(Float64,x) for x in split(readline())]
    #z1i=parse(Float64,readline())
    println("Enter second complex number...")
    z2r,z2i=[parse(Float64,x) for x in split(readline())]
    #z2r,z2i=parse(Float64,readline())
    #z2i=parse(Float64,readline())
    z1=complex(z1r,z1i)
    z2=complex(z2r,z2i)
    z3=z1 + z2
    z4=z1-z2
    z5=z1*z2
    z6=z1/z2
    println("Sum=$z3\nDiff=$z4\nProduct=$z5\nQuotient=$z6")
end
complexarithmetic()
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter first complex number...
4 5
Enter second complex number...
6 9
Sum=10.0 + 14.0im
Diff=-2.0 - 4.0im
Product=-21.0 + 66.0im
Quotient=0.5897435897435898 - 0.05128205128205131im
```

1c. Develop a Julia program to evaluate expressions having mixed data types (integer, real, floating-point.

```
function evaluatemixed()
    println("Enter a expression with mixed types")
    expr = readline()
    println(eval(expr))
end
evaluatemixed()
```

Output

```
Enter a expression with mixed types
julia> 4 + 3/2.1 - 5.6 + 3im - 4im


julia> -0.17142857142857082 - 1.0im

julia>
```

2a. Develop a Julia program for the following problem: A computer repair shop charges $100 per hour for labour plus the cost of any parts used in the repair. However, the minimum charge for any job is $150. Prompt for the number of hours worked and the cost of parts (which could be $0) and print the charge for the job.

```
function makepay()
    print("Enter the number of hours you want to pay")
    hours = parse(Int,readline())
    print("Cost of parts")
    cost = parse(Int,readline())
    tot = max(100*hours+cost,150)
    println(tot)
end
makepay()
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter the number of hours you want to pay 45
Cost of parts 20
4520
```

2b. Develop a Julia program to calculate a person's regular pay, overtime pay and gross pay based on the following: If hours worked is less than or equal to 40, regular pay is calculated by multiplying hours worked by rate of pay, and overtime pay is 0. If hours worked is greater than 40, regular pay is calculated by multiplying 40 by the rate of pay, and overtime pay is calculated by multiplying the hours in excess of 40 by the rate of pay by 1.5. Gross pay is calculated by adding regular pay and overtime pay.

```
function computer_pay()
    print("Enter the number of hours ")
    hours = parse(Int,readline())
    print("Rate of pay")
    rate=parse(Int,readline())
    if hours<=40
        regularpay=hours*rate
        overtimepay=0
    elseif hours>40
        regularpay=40*rate
        overtimepay=(hours-40)*rate*1.5
    end
    grosspay = regularpay+overtimepay
    println("Regular
pay=$regularpay,Overtimepay=$overtimepay,grosspay=$grosspay")
end
computer_pay()
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter the number of hours 39
Rate of pay 100
Regular pay=3900,Overtimepay=0,grosspay=3900
```

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter the number of hours 45
Rate of pay 100
Regular pay=4000,Overtimepay=750.0,grosspay=4750.0
```

3a. An amount of money P (for principal) is put into an account which earns interest at r% per annum. So, at the end of one year, the amount becomes P + P×r/100. This becomes the principal for the next year. Develop a Julia program to print the amount at the end of each year for the next 10 years. However, if the amount ever exceeds 2P, stop any further printing. Your program should prompt for the values of P and r.

```
function print_principal()
    print("Enter P")
    P=parse(Int,readline())
    print("Enter r")
    r=parse(Float64,readline())
    i=1
    OP=P
    while i<=10
      P=P+(P*r)/100
      if P>=2*OP
        break
      end
      println(P)
      i=i+1
    end
end
print_principal()
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter P 100
Enter r 60
160.0
```

3b. Develop a Julia program which reads numbers from a file (input.txt) and finds the largest number, smallest number, count, sum and average of numbers.

```
function find_number_details()
    largest = 0
    smallest = 100000000000000000
    count = 0
```

```
    sum = 0
    for num in eachline("inp.txt")
        num = parse(Int,num)
        if num > largest
            largest = num
        end
        if num < smallest
            smallest = num
        end
        sum = sum + num
        count = count + 1
    end
    avg = sum/count
    println("Largest = $largest\nSmallest = $smallest\nCount = $count\nSum = $sum\nAverage = $avg")
end
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Largest = 78
Smallest = 10
Count = 6
Sum = 220
Average = 36.666666666666664
```

4a. Develop a Julia program and two separate functions to calculate GCD and LCM.

```
function gcd(m,n)
    while n!=0
        r=m%n
        m=n
        n=r
    end
    return m
```

```
end
function lcm(m,n)
    return m*n/gcd(m,n)
end
while true
println("Enter the value of m and n")
m,n=[parse(Int64, x) for x in split(readline())]
if m<=0 || n<=0
    break
end
println("GCD=", gcd(m,n))
println("Lcm=", lcm(m,n))
end
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter the value of m and n
22 48
GCD=2
Lcm=528.0
```

4b. Develop a Julia program and a recursive function to calculate factorial of a number.

```
function factorial(n)
    if n<=1
        return 1
    end
    return n*factorial(n-1)
end
println("Enter a number")
n=parse(BigInt,readline())
println(factorial(n))
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter a number
10
3628800
```

4c. Develop a Julia program and a recursive function to generate Fibonacci series.

```
function fibonacc(n)
    if n==0 || n==1
        return 1
    end
    return fibonacc(n-1) + fibonacc(n-2)
end
println("Enter n")
term = parse(Int, readline())
println(fibonacc(term-1))
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter n
15
610
```

5a. Develop a Julia program which reads a string (word) and prints whether the word is palindrome.

```
function isPalindrome(word)
    if lowercase(word) == lowercase(reverse(word))
        println("is a palindrome")
    else
        println("not a palindrome")
    end
end
println("enter a word")
word=readline()
```

```
isPalindrome(word)
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
enter a word
Malayalam
is a palindrome
```

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
enter a word
jnnce
not a palindrome
```

5b. Develop a Julia program which reads and prints the words present in a file (input.txt) having Random Data in which words are dispersed randomly (Assumption: a word is a contiguous sequence of letters. A word is delimited by any non-letter character or end-of-line).

```
function extractwordsfromfile()
    mydata = read("input.txt", String)
    word=""
    for ch in mydata
        if isletter(ch)
            word=word*ch
        else
            print(word," ")
            word=""
        end
    end
end
extractwordsfromfile()
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
My aim is to become successfull person  Terminal will be reused by tasks, press any key to close it.
```

6a. Develop a Julia program to determine and print the frequency with which each letter of the alphabet is used in a given line of text.

```
function findletterfrequencies(line)
    freq=fill(0,26)
    line=lowercase(line)
    for ch in line
        if isletter(ch)
            freq[ch-'a'+1] += 1
        end
    end
    for ch='a':'z'
        println("Frequency of $ch is ",freq[ch-'a'+1])
    end
end
println("Enter a line of text")
line=readline()
findletterfrequencies(line)
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter a line of text
Aiml jnnce
Frequency of a is 1
Frequency of b is 0
Frequency of c is 1
Frequency of d is 0
Frequency of e is 1
Frequency of f is 0
Frequency of g is 0
Frequency of h is 0
Frequency of i is 1
Frequency of j is 1
Frequency of k is 0
Frequency of l is 1
Frequency of m is 1
Frequency of n is 2
Frequency of o is 0
Frequency of p is 0
Frequency of q is 0
Frequency of r is 0
Frequency of s is 0
Frequency of t is 0
Frequency of u is 0
Frequency of v is 0
Frequency of w is 0
Frequency of x is 0
Frequency of y is 0
Frequency of z is 0
```

6b. A survey of 10 pop artists is made. Each person votes for an artist by specifying the number of the artist (a value from 1 to 10). Develop a Julia program to read the names of the artists, followed by the votes, and find out which artist is the most popular.

```
function voting()
    println("Enter number of candidates")
    n=parse(Int,readline())
    println("Enter number of votes")
    m=parse(Int,readline())
    candidates=fill("",n)
    votes=fill(0,n)
    for i in 1:n
        println("Enter candidate $i")
        candidates[i]=readline()
    end
    for i = 1:m
        println("Enter vote $i")
        vote=parse(Int,readline())
        if vote>=0 && vote<=n
            votes[vote]+=1
        end
    end
    maxvote=maximum(votes)
    println("maxvote is $maxvote")
    for i in 1:n
        if votes[i]==maxvote
            println(candidates[i])
        end
    end
end
voting()
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter number of candidates
3
Enter number of votes
10
Enter candidate 1
Aaliya
Enter candidate 2
Ashwini
Enter candidate 3
Shaziya
Enter vote 1
4
Enter vote 2
3
Enter vote 3
5
Enter vote 4
2
Enter vote 5
1
Enter vote 6
2
Enter vote 7
1
Enter vote 8
4
Enter vote 9
7
Enter vote 10
2
maxvote is 3
Ashwini
```

7a. Given a line of text as input, develop a Julia program to determine the frequency with which each letter of the alphabet is used (make use of dictionary)

```
function findletterfrequencies(line)
    freq=Dict{Char,Int}()
    line=lowercase(line)
    for ch in line
        if isletter(ch)
            freq[ch]=get(freq,ch,0)+1
        end
    end
    for ch in sort(collect(keys(freq)))
        println("Frequency of $ch is ",freq[ch])
    end
end
println("Enter a line of text")
```

```
line=readline()

findletterfrequencies(line)
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter a line of text
Hello JNNCE
Frequency of c is 1
Frequency of e is 2
Frequency of h is 1
Frequency of j is 1
Frequency of l is 2
Frequency of n is 2
Frequency of o is 1
```

7b. Develop a Julia program to fetch words from a file with arbitrary punctuation and keep track of all the different words found (make use of set and ignore the case of the letters: e.g. to and To are treated as the same word).

```
function returnuniquewords()
    mydata = read("inp1.txt",String)
    mydata = lowercase(mydata)
    wordlist = Set{String}()
    word =""
    for ch in mydata
        if isletter(ch)
            word = word*ch
        else
            if word!= ""
                union!(wordlist,[word])
                word=""
            end
        end
    end
    return wordlist
end
println(returnuniquewords())
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Set(["you", "not", "were", "and", "happy", "should", "be", "felt", "extremely", "but", "saw", "because", "was", "knew", "really", "are", "others", "i"])
  Terminal will be reused by tasks, press any key to close it.
```

8a. Develop a Julia program to evaluate expressions consisting of rational, irrational number and floating-point numbers)

```
function evaluate_expression(expr)
    try
        result = eval(Meta.parse(expr))
        return result
    catch e
        return "Error: $(e)"
    end
end
println("Enter an expression to evaluate")
iexpr = readline()
println(evaluate_expression(iexpr))
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
Enter an expression to evaluate
1/2 + pi + 0.6
4.241592653589793
```

8b. Develop a Julia program to determine the following properties of a matrix: determinant, inverse, rank, upper & lower triangular matrix, diagonal elements, Euclidean norm and Square Root of a Matrix.

```
using LinearAlgebra
using Pkg
import Pkg
Pkg.add("PrettyTables")
using PrettyTables
```

```
function matrixfunctions(A)
    println(det(A))
    pretty_table(inv(A))
    C = [1 2 3;2 4 6;1 5 7]
    println(rank(C))
    pretty_table(triu(A))
     pretty_table(tril(A))
     pretty_table(diag(A))
    println(norm(A))
    B = [5 4; 4 5]
    pretty_table(sqrt(B))
end
A = [1 2 3; 1 4 1; 2 1 3]
pretty_table(A)
matrixfunctions(A)
```

Output

Col. 1	Col. 2	Col. 3
1	2	3
1	4	1
2	1	3

-12.0

Col. 1	Col. 2	Col. 3
-0.916667	0.25	0.833333
0.0833333	0.25	-0.166667
0.583333	-0.25	-0.166667

2

Col. 1	Col. 2	Col. 3
1	2	3
0	4	1
0	0	3

Col. 1	Col. 2	Col. 3
1	0	0
1	4	0
2	1	3

Col. 1
1
4
3

6.782329983125268

Col. 1	Col. 2
2.0	1.0
1.0	2.0

9a. Develop a Julia program to determine addition and subtraction of two matrices (element -wise).

```
using LinearAlgebra

function matrxiop(A,B)

    C=A+B

    display(C)

    D=A-B

    display(D)

end

A=[1 2 3;1 4 1;2 1 3]

B=[1 0 -2;5 4 0;2 1 1]

display(A)

display(B)

matrxiop(A,B)
```

Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
3×3 Matrix{Int64}:
 1  2  3
 1  4  1
 2  1  3
3×3 Matrix{Int64}:
 1  0  -2
 5  4   0
 2  1   1
3×3 Matrix{Int64}:
 2  2  1
 6  8  1
 4  2  4
3×3 Matrix{Int64}:
  0  2  5
 -4  0  1
  0  0  2
```

9b. Develop a Julia program to perform multiplication operation on matrices: scalar multiplication, element-wise multiplication, dot product, cross product.

```
using LinearAlgebra

function matrixop(A,B)

    C=2*A

    display(C)

    D=A.*B

    display(D)

    E=dot(A,B)

    display(E)

    A=[1,2,3]

    B=[4,5,6]

    F=cross(A,B)

    display(F)

end

A=[1 3;4 2]

B=[1 6;2 1]

matrixop(A,B)
```

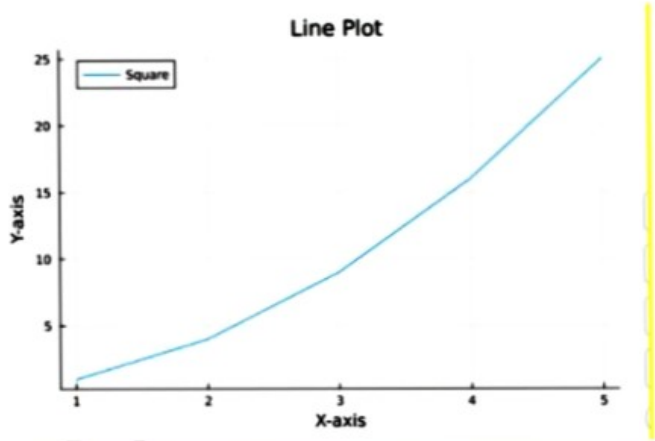
Output

```
  Activating project at `C:\Users\MBA\.julia\environments\v1.11`
2×2 Matrix{Int64}:
 2  6
 8  4
2×2 Matrix{Int64}:
 1  18
 8   2
29
3-element Vector{Int64}:
 -3
  6
 -3
```

10a. Develop a Julia program to generate a plot of (solid & dotted) a function: $y=x^2$ (use suitable data points for x).

```
import Pkg
Pkg.add("Plots")
using Plots
x = Array([1,2,3,4,5])
y=x.^2
plot(x, y, label="Square", xlabel="X-axis", ylabel="Y-axis", title="Line Plot")
# Save the plot to a file
savefig("line_plot.png")
```

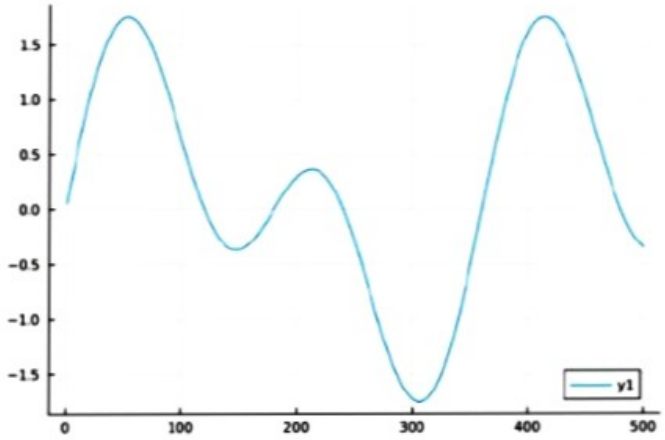
Output



10b. Develop a Julia program to generate a plot of mathematical equation: y = sin(x) + sin(2x).

```
using Plots
eq(x) = sind(x) + sind(2x)
plot(eq, 1:500)
savefig("sind_plot")
```

Output



10c. Develop a Julia program to generate multiple plots of mathematical equations: y = s in(x) + sin(2x) and y

```
using Plots

eq(x) = sind(x) + sind(2x)

plot(eq, 1:500)

eq1(x) = sind(x) + sind(3x)

plot!(eq1, 1:500)

savefig("sind_new_plot")
```

Output