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

```
function calculator()
    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(calculator())
```

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()
```

1c) Develop a Julia program to evaluate expressions having mixed datatypes (integer, real, floating-point number and complex).

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

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 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()
```

2b) Develop a Julia program to calculate a person's regular pay, overtime pay and gross pay bases 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("Regularpay=$regularpay,Overtimepay=$overtimepay,grosspay=$grosspay")
end
computer_pay()
```

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*/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()
```

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("input.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
find_number_details()
```

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(&amp;quot;Enter the value of m and n&amp;quot;)
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
```

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))
```

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 ))
```

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)
```

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()
```

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)
```

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()
```

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 linebof text")
```

```
line=readline()
findletterfrequencies(line)
```

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: eg 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())
```

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(expr))
                        return result
                catch e
                        return "Error: $(e)"
                end
        end
        println("Enter an expression to evaluate")
        iexpr = readline()
        println(evaluate_expression(iexpr)
```

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("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)
```

9a) Develop a Julia program to determine addition and substraction 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)
```

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)
```

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")
savefig("line_plot.png")
```

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")
```

10c)Develop a Julia program to generate multiple plots of mathematical equations: y=sin(x)+sin(2x) and y=sin(2x)+sin(3x)

```
using Plots
eq(x)=sind(x)+sind(2x)
plot(eq,1:500)
eq1(x)=sind(2x)+sind(3x)
plot!(eq,1:500)
savefig("sind_new_plot")
```