

LAB PROGRAM 1A

```
function display_menu()
    println("Welcome to trhe calculator Program")
    println("1.Addition")
    println("2.Subtraction")
    println("3.Multiplication")
    println("4.Division")
    println("5. Exit")
    println("Enter your choice(1-5):")
end

function addition(a,b)
    return a+b
end
function subtraction(a,b)
    return a-b
end
function multiplication(a,b)
    return a*b
end
function division(a,b)
    if b!=0
        return a/b
    else
        println("Error:Division by zero!")
        return NaN
    end
end

function main()
    while true
        display_menu()
        choice=parse{Int64,readline()}
        if choice==5
            println("Exiting Calculator Program.")
            break
        end
        println("Enter first number:")
        num1=parse{Float64,readline()}
        println("Enter second number:")
        num2=parse{Float64,readline()}

        if choice==1
            result=addition(num1,num2)
            println("Result:",result)
        elseif choice==2
            result=subtraction(num1,num2)
```

```

        println("Result:",result)
elseif choice==3
    result=multiplication(num1,num2)
    println("Result:",result)
elseif choice==4
    result=division(num1,num2)
    println("Result:",result)
else
    println("Invalid choice! Please enter a number between 1 and 5.")
end
end
end

main()

```

LAB PROGRAM 1B

```

struct ComplexNumber
    real::Float64
    imag::Float64
end

```

```

function add_complex(a::ComplexNumber, b::ComplexNumber)
    return ComplexNumber(a.real + b.real, a.imag + b.imag)
end

```

```

function subtract_complex(a::ComplexNumber, b::ComplexNumber)
    return ComplexNumber(a.real - b.real, a.imag - b.imag)
end

```

```

function multiply_complex(a::ComplexNumber, b::ComplexNumber)
    real_part = a.real * b.real - a.imag * b.imag
    imag_part = a.real * b.imag + a.imag * b.real
    return ComplexNumber(real_part, imag_part)
end

```

```

function divide_complex(a::ComplexNumber, b::ComplexNumber)
    denominator = b.real^2 + b.imag^2
    if denominator != 0
        real_part = (a.real * b.real + a.imag * b.imag)
        imag_part = (a.imag * b.real - a.real * b.imag)
        return ComplexNumber(real_part, imag_part)
    else

```

```

        println("Error: Division by zero!")
        return ComplexNumber(NaN, NaN)
    end
end

function main()
    println("Enter the real and imaginary parts of the first complex number:")
    real1 = parse(Float64, readline())
    imag1 = parse(Float64, readline())
    println("Enter the real and imaginary parts of the second complex number:")
    real2 = parse(Float64, readline())
    imag2 = parse(Float64, readline())

    complex1 = ComplexNumber(real1, imag1)
    complex2 = ComplexNumber(real2, imag2)

    println("Addition: ", add_complex(complex1, complex2))
    println("Subtraction: ", subtract_complex(complex1, complex2))
    println("Multiplication: ", multiply_complex(complex1, complex2))
    println("Division: ", divide_complex(complex1, complex2))
end

```

```
main()
```

LAB PROGRAM 1C

```

function evaluate_expression(expression)
    try
        result=Meta.parse(expression)
    println("Result:",eval(result))
    catch e
    println("Error:",e)
    end
end

function main()
    println("Enter the expression to evaluate:")
    expression = readline()
    evaluate_expression(expression)
end
main()

```

LAB PROGRAM 2A

```
using Printf
function jobCharge()
    print("Hours worked?")
    hours=parse(Float64,readline())
    print("Cost of parts?")
    parts=parse(Float64,readline())
    jobCharge=hours*100+parts
    if jobCharge < 150
        jobCharge = 150
    end
    @printf("\n Total charges:\$%0.2f\n",jobCharge)
end
jobCharge()
```

LAB PROGRAM 2B

```
using Printf
function calculatePay()
    print("Hours Worked?")
    hours=parse(Float64,readline())
    print("Rate of pay?")
    rate=parse(Float64,readline())
    if hours<=40
        regPay=hours*rate
        ovtPay=0
    else
        regPay=40*rate
        ovtPay=(hours-40)*rate*1.5
    end
    grossPay=regPay+ovtPay
    @printf("\nRegular pay:\$%0.2f\n",regPay)
    @printf("Overtime pay:\$%03.2f\n",ovtPay)
    @printf("Gross pay:\$%0.2f\n",grossPay)
end
calculatePay()
```

LAB PROGRAM 3A

```
using Printf
function calcInterest()
print("Principal?")
P=parse{Int64,readline()}
print("Interest Rate?")
r=parse{Float64,readline()}
println("Year Amount")
amt=P
for y =1:10
amt+= amt*r/100
@printf("%3d%8.2f\n",y,amt)
if amt> 2P break end
end
end
calcInterest()
```

LAB PROGRAM 3B

```
function analyze_numbers(file_name::String)
if !isfile(file_name)
    println("Error:File not found!")
    return
end
numbers=[]
open(file_name,"r")do file
    for line in eachline(file)
        push!(numbers,parse{Float64,strip(line)})
    end
end

largest=maximum(numbers)
smallest=minimum(numbers)
count=length(numbers)
total=sum(numbers)
average=total/count

    println("Analysis of numbers in file:")
    println("largest number:$largest")
    println("Smallest numbere:$smallest")
    println("Count of number:$count")
    println("Sum of numbers:$total")
    println("Average of numbers:$average")
end

file_name="input.txt"
analyze_numbers(file_name)
```

