SCIENTIFIC CALCULATOR

21CSS101J – PROGRAMMING FOR PROBLEM-SOLVING

Mini Project Report

TIRUVAIPATI V N S K MOHAN [Reg. No.: RA2311003011004]
B.Tech. CSE - CORE

SCHOOL OF COMPUTING
COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603 203
CHENGALPATTU DISTRICT

November 2023

COLLEGE OF ENGINEERING AND TECHNOLOGY
SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
(Under Section 3 of UGC Act, 1956)
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that Mini project report titled _____ is the bonafide work of Reg.No_____Name_____who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
(GUIDE)

SIGNATURE
(HEAD OF THE DEPARTMENT)

TABLE OF CONTENTS

S No.	Title	Page No.
1	Problem Statement	
2	Methodology / Procedure/ Algorithm	
3	Flowchart	
4	Coding (C/Python)	
5	Modules of the proposed work	
6	Results/Screenshots	
7	Conclusion	
8	References	

1.Problem Statement

In the realm of scientific computation, there exists a growing demand for a versatile and sophisticated scientific calculator that goes beyond basic arithmetic operations. The objective of this project is to design and implement an advanced scientific calculator capable of handling a diverse range of mathematical functions. The calculator should encompass functionalities such as arithmetic operations, trigonometric calculations, exponential functions, algebraic manipulations, matrix operations, unit conversions, and access to essential mathematical constants.

Challenges to Address:

Comprehensive Functionality:

Develop a user-friendly interface that accommodates a wide array of mathematical functions, providing users with a seamless experience in performing arithmetic, trigonometric, exponential, algebraic, and matrix operations.

Accurate Trigonometric Computations:

Implement precise algorithms for trigonometric functions to ensure accuracy in calculations, taking into consideration angles in degrees and radians.

Efficient Exponential Calculations:

Optimize exponential calculations, including exponentiation and logarithmic functions, to handle a broad range of input values while maintaining computational efficiency.

Algebraic Manipulations:

Enable the calculator to perform algebraic operations such as solving equations, simplifying expressions, and handling variables, enhancing its utility for students and professionals alike.

Matrix Operations and Conversions:

Incorporate matrix operations, including addition, subtraction, multiplication, and inversion, along with the capability to convert between different matrix representations.

Unit Conversions:

Provide a mechanism for seamless unit conversions across various measurement systems, catering to the needs of users in different scientific disciplines.

Access to Constants:

Include a comprehensive library of essential mathematical constants, making them readily accessible for users engaged in advanced scientific and engineering calculations.

The successful development of such an advanced scientific calculator would contribute significantly to the scientific and educational communities, providing a powerful tool for professionals, students, and researchers engaged in diverse mathematical disciplines.

2.Methodology / Procedure/ Algorithm

Start: The program begins.

Select an operation: The user selects an function like
arithmetic operation (+, -, *, /).
Trigonometric (sin, cos, tan, inverse of sin cos tan)
Exponential (e*x , logx)
Algebraic (linear, quadratic)
Matrix (trace, det, multiplication)
Unit conversions (time, length, weight)
Constants

Input a number: The user enters a number.

Selection of operation: now the user specifies the wanted operation in that function

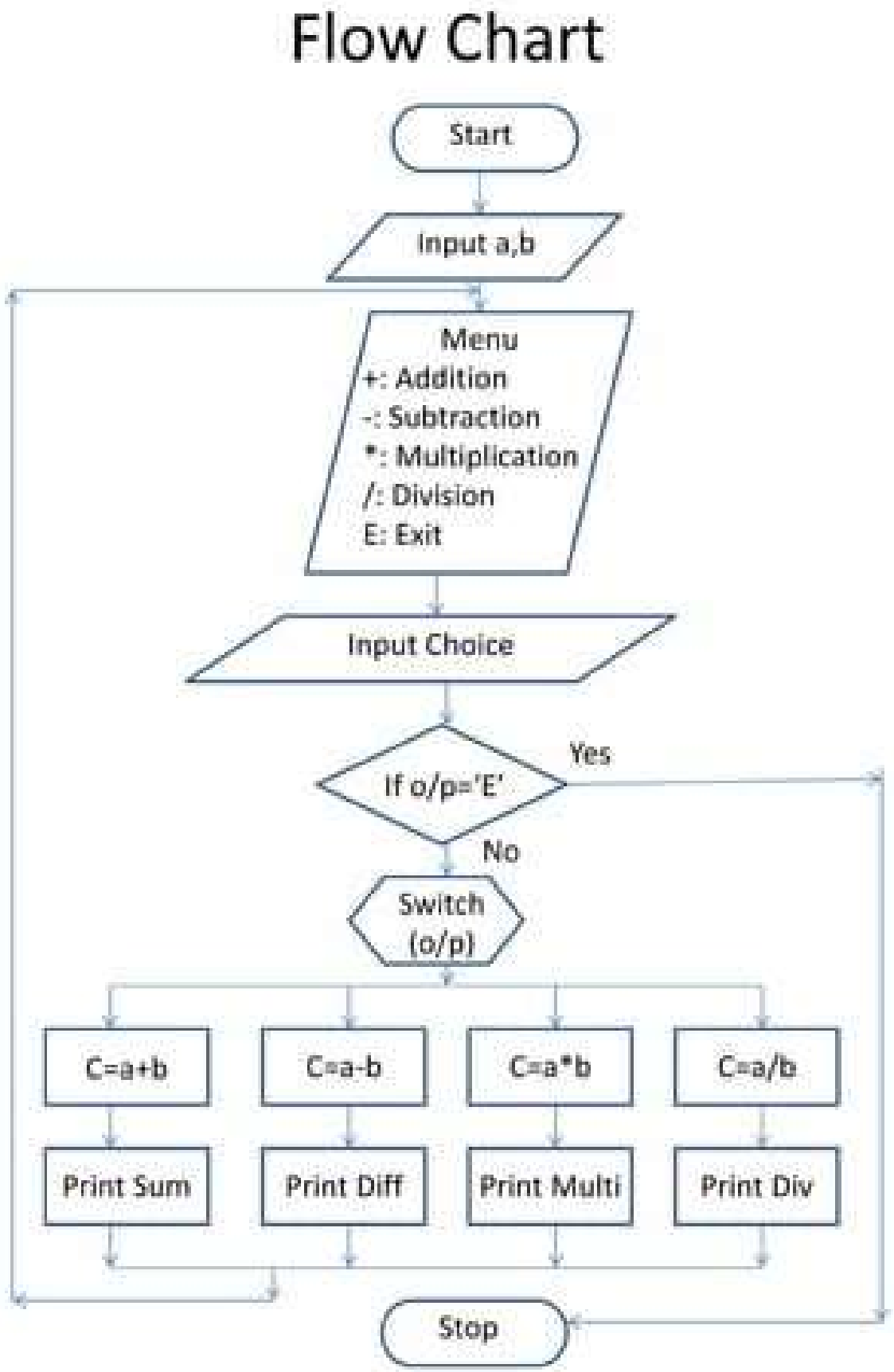
Input another number: The user enters another number.

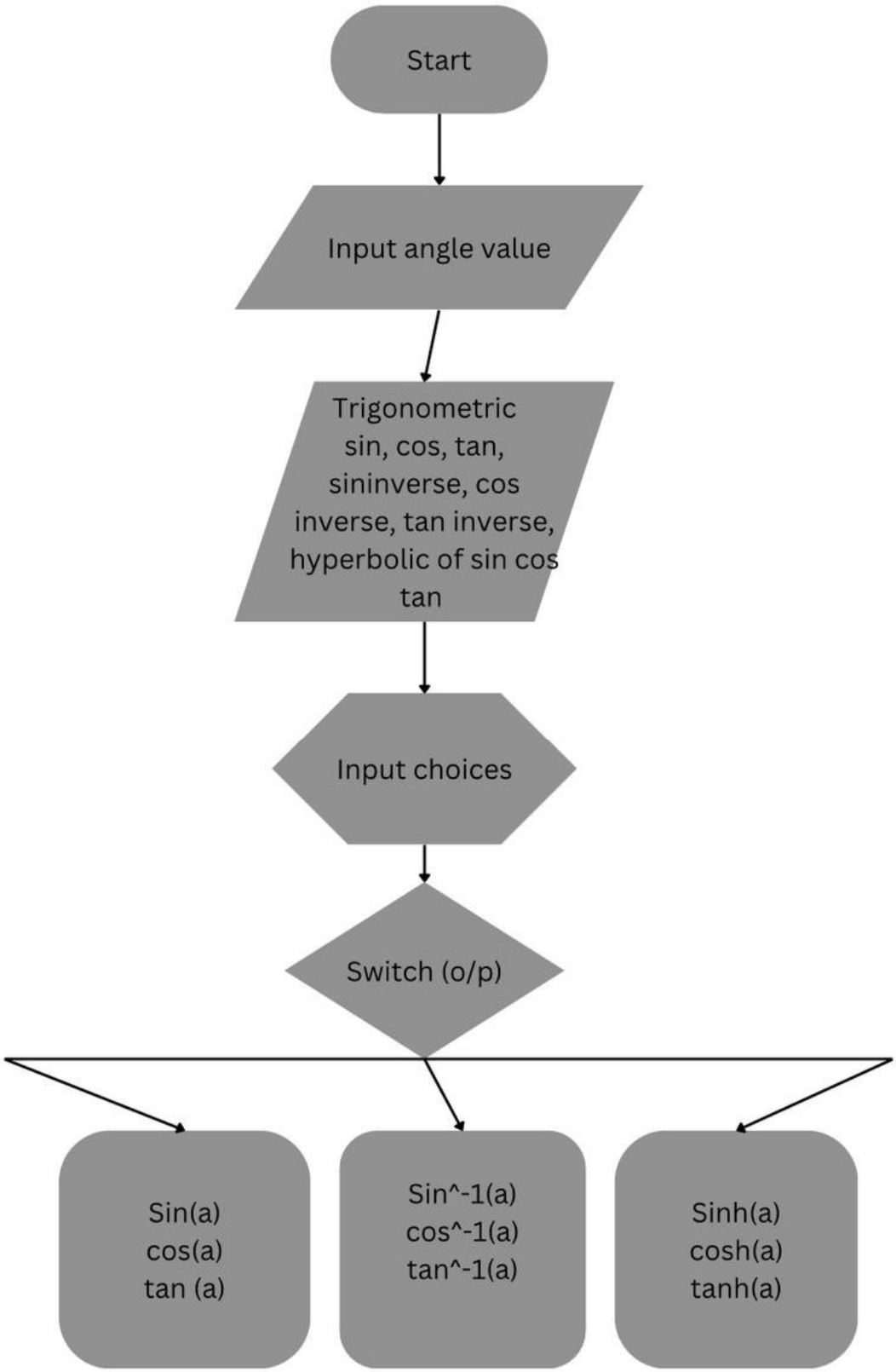
Perform the selected operation: The calculator performs the selected operation on the two input numbers.

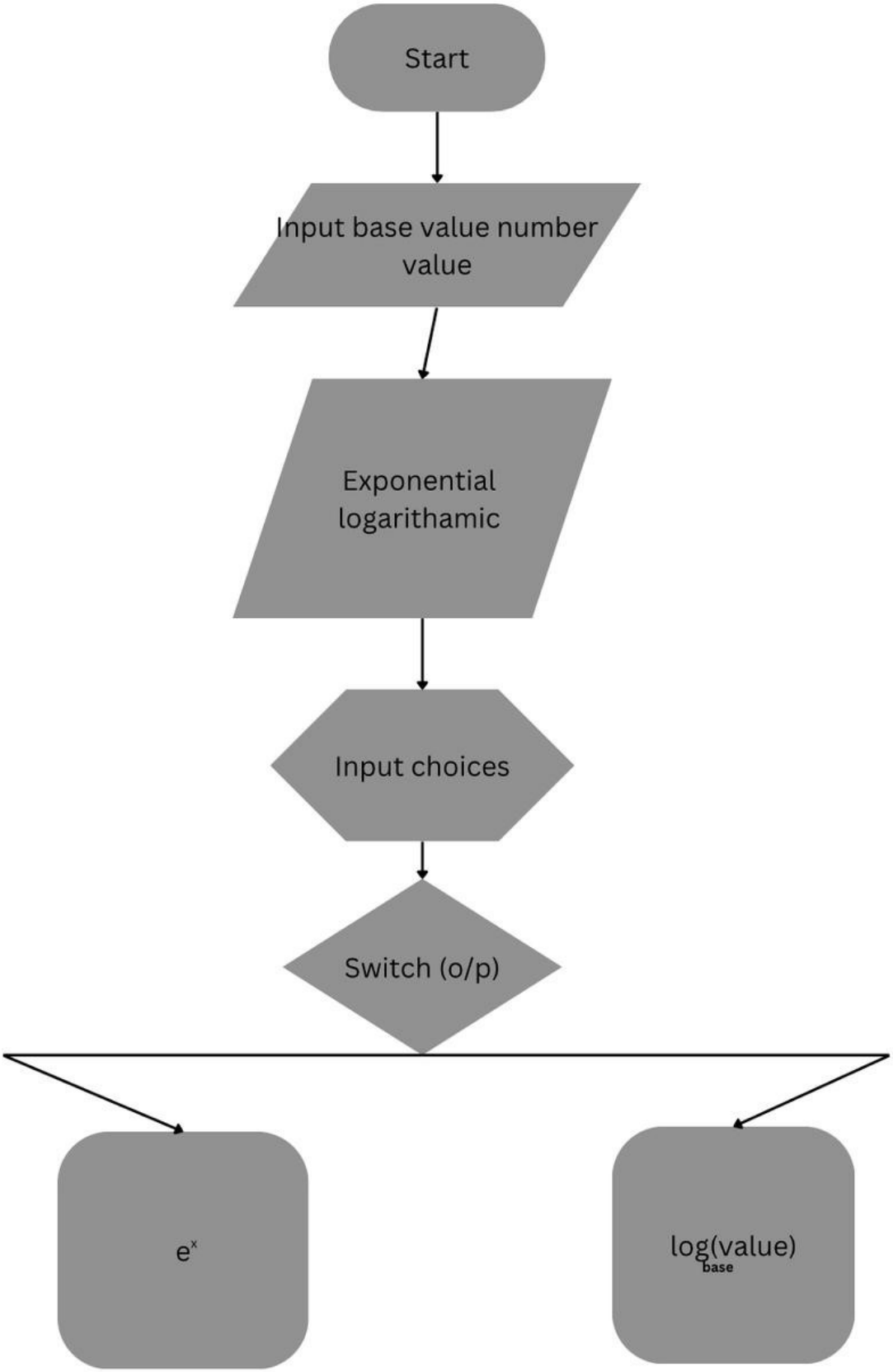
Display the result: The calculator displays the result to the user.

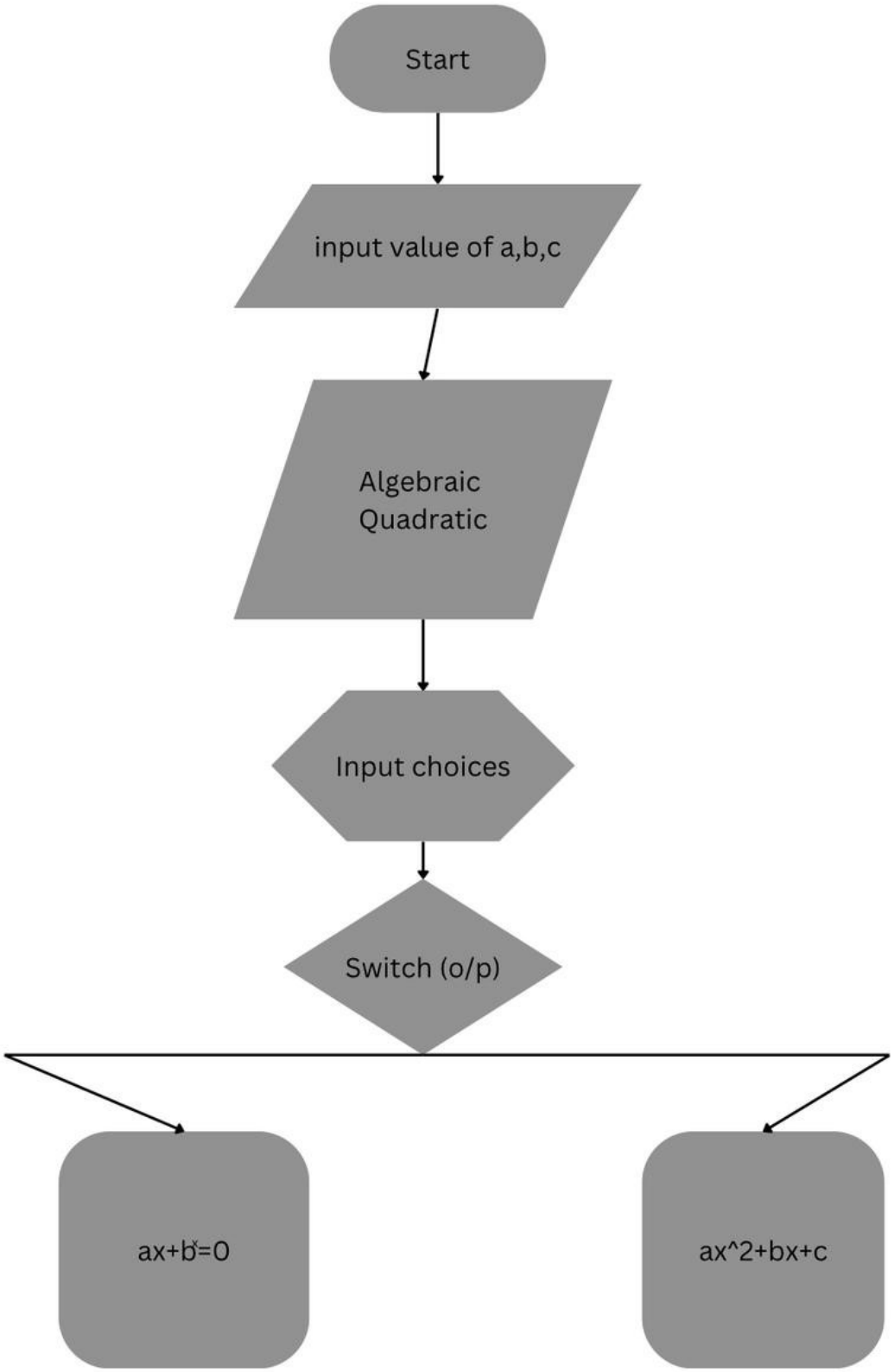
Repeat (**Y/N):** The user is asked if they want to perform another calculation. If "Yes," the program goes back to step 2. If "No," the program ends.

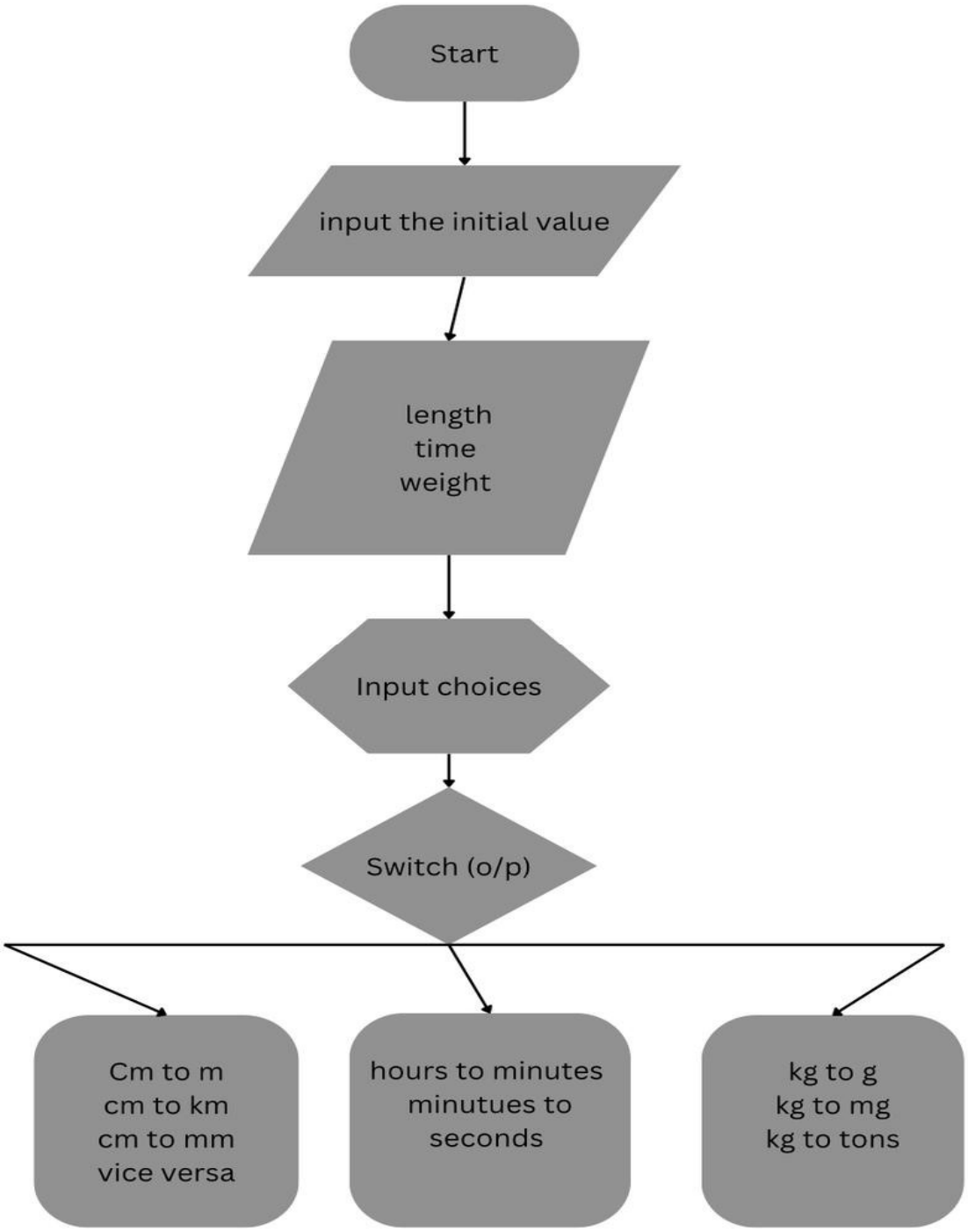
3. Flow chart:

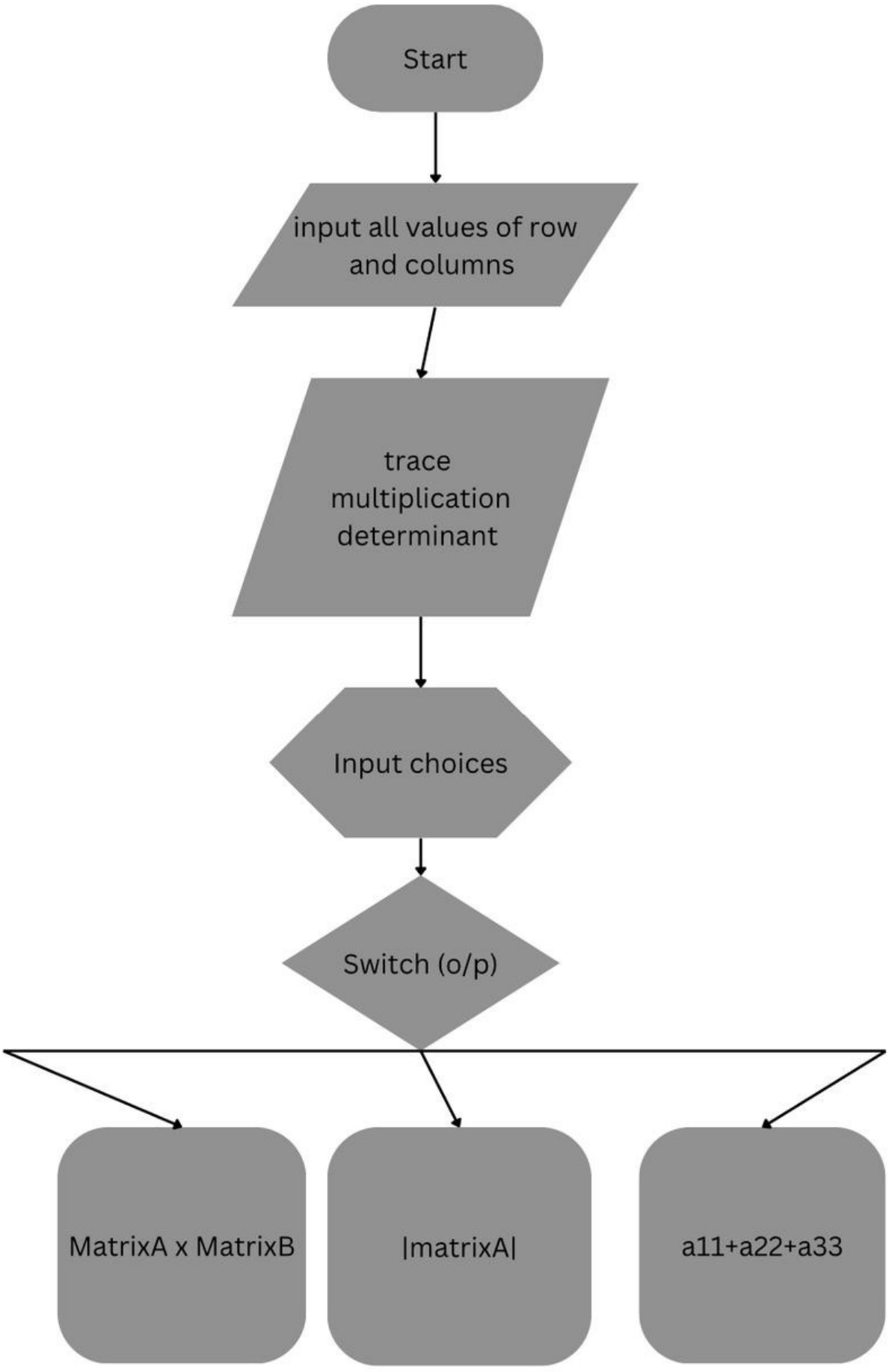

Flow Chart
Start
Input a,b
Menu
+: Addition
-: Subtraction
*: Multiplication
/: Division
E: Exit
Input Choice
If o/p='E'
Yes
No
Switch (o/p)
C=a+b
C=a-b
C=a*b
C=a/b
Print Sum
Print Diff
Print Multi
Print Div
Stop


Start
Input angle value
Trigonometric sin, cos, tan, sininverse, cos inverse, tan inverse, hyperbolic of sin cos tan
Input choices
Switch (o/p)
Sin(a)
cos(a)
tan (a)
Sin^-1(a)
cos^-1(a)
tan^-1(a)
Sinh(a)
cosh(a)
tanh(a)


Start
Input base value number value
Exponential logarithamic
Input choices
Switch (o/p)
e^x
$\log_{base}(value)$


Start
input value of a,b,c
Algebraic
Quadratic
Input choices
Switch (o/p)
ax+b=0
ax^2+bx+c


Start
input the initial value
length
time
weight
Input choices
Switch (o/p)
Cm to m
cm to km
cm to mm
vice versa
hours to minutes
minutues to
seconds
kg to g
kg to mg
kg to tons


Start
input all values of row and columns
trace multiplication determinant
Input choices
Switch (o/p)
MatrixA x MatrixB
|matrixA|
a11+a22+a33

4. Coding (C/Python):

```
#include <stdio.h>
#include <math.h>

int main() {
   int choice;
   double num1, num2, result;
    double angle;
   int i,opt;
    double x, base;
int fromUnit, toUnit;
   double value;
   double temperature;
    double a, b,c;
    int rows, cols;
     double matrix1[rows][cols];
   double matrix2[rows][cols];
   double resultm[rows][cols];
printf("\n");
for(i=0;i<=120;i++)
{
   printf("-");
}
printf("\n");




   printf("Select a mathematical operation:\n");
   printf("1. Arithmetic\n2. Trigonometric\n3. Exponential\n4. Algebraic\n5.Conversions\n6.
Constants\n7. Matrix \n");
   printf("enter choice : ");
   scanf("%d", &choice);

   printf("\n");
for(i=0;i<=120;i++)
{
   printf("-");
}
printf("\n");
void displayMatrix(int rows, int cols, double matrix[rows][cols]) {
```

```
    printf("\nMatrix:\n");
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            printf("%.2f\t", matrix[i][j]);
        }
        printf("\n");
    }
}

// Function to perform matrix addition
void matrixAddition(int rows, int cols, double matrix1[rows][cols], double matrix2[rows][cols], double resultm[rows][cols]) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            resultm[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }
}

// Function to perform matrix subtraction
void matrixSubtraction(int rows, int cols, double matrix1[rows][cols], double matrix2[rows][cols], double resultm[rows][cols]) {
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            resultm[i][j] = matrix1[i][j] - matrix2[i][j];
        }
    }
}

// Function to perform matrix multiplication
void matrixMultiplication(int rows1, int cols1, int rows2, int cols2, double matrix1[rows1][cols1], double matrix2[rows2][cols2], double resultm[rows1][cols2]) {
    for (int i = 0; i < rows1; ++i) {
        for (int j = 0; j < cols2; ++j) {
            resultm[i][j] = 0;
            for (int k = 0; k < cols1; ++k) {
                resultm[i][j] += matrix1[i][k] * matrix2[k][j];
            }
        }
    }
}

// Function to calculate the trace of a matrix
```

```
double matrixTrace(int rows, int cols, double matrix[rows][cols]) {
   double trace = 0;
   for (int i = 0; i < rows && i < cols; ++i) {
      trace += matrix[i][i];
   }
   return trace;
}

// Function to calculate the determinant of a 2x2 matrix
double matrixDeterminant2x2(double matrix[2][2]) {
   return matrix[0][0] * matrix[1][1] - matrix[0][1] * matrix[1][0];
}

// Function to calculate the determinant of a 3x3 matrix
double matrixDeterminant3x3(double matrix[3][3]) {
   return matrix[0][0] * (matrix[1][1] * matrix[2][2] - matrix[1][2] * matrix[2][1]) -
          matrix[0][1] * (matrix[1][0] * matrix[2][2] - matrix[1][2] * matrix[2][0]) +
          matrix[0][2] * (matrix[1][0] * matrix[2][1] - matrix[1][1] * matrix[2][0]);
}

// Function to calculate the determinant of a matrix
double matrixDeterminant(int rows, int cols, double matrix[rows][cols]) {
   if (rows == 2 && cols == 2) {
      return matrixDeterminant2x2(matrix);
   } else if (rows == 3 && cols == 3) {
      return matrixDeterminant3x3(matrix);
   } else {
      printf("Determinant calculation not implemented for matrices larger than 3x3.\n");
      return 0.0;
   }
}
   switch (choice)
   {
      case 1: // Arithmetic
         printf("Enter two numbers: ");
         scanf("%lf %lf", &num1, &num2);
         printf("Select an operation:\n1. Addition\n2. Subtraction\n3. Multiplication\n4.
Division\n5. Fraction\n6. Decimals\n7. Percentage\n8. Factorial\n9. Summation\n");
         scanf("%d", &choice);

         switch (choice) {
            case 1:
               result = num1 + num2;
```

```
            printf("%d",result);
            break;
        case 2:
            result = num1 - num2;
             printf("%d",result);
            break;
        case 3:
            result = num1 * num2;
             printf("%d",result);
            break;
        case 4:
            result = num1 / num2;
             printf("%d",result);
            break;


            return 1;
        }
        break;
printf("\n");
for(i=0;i<=120;i++)
{
   printf("-");
}
printf("\n");
      case 2: // Trigonometric
         // Similar structure for trigonometric functions
          // Input angle in degrees


   printf("Enter an angle in degrees: ");
   scanf("%lf", &angle);

   // Convert angle to radians
   double angle_rad = angle * M_PI / 180.0;
   printf("\n");
for(i=0;i<=120;i++)
{
   printf("-");
}
printf("\n");
   printf("\nTrigonometric functions:\n");
```

```
   printf("\n option 1 sin ");
    printf("\n option 2 cos");
     printf("\n option 3 tan");
     printf("\nInverse trigonometric functions:\n");
      printf("\n option 4 sin inverse");
       printf("\n option 5 cos inverse");
        printf("\n option 6 tan inverse");

   printf("\nHyperbolic functions:\n");
        printf("\n option 7 sinh");
         printf("\n option 8 cosh");
          printf("\n option 9 tanh");

          printf("\n enter the option number : ");
          scanf("%d",&opt);
   // Trigonometric functions
   double sin_val = sin(angle_rad);
   double cos_val = cos(angle_rad);
   double tan_val = tan(angle_rad);

   // Inverse trigonometric functions
   double asin_val = asin(sin_val);
   double acos_val = acos(cos_val);
   double atan_val = atan(tan_val);

   // Hyperbolic functions
   double sinh_val = sinh(angle_rad);
   double cosh_val = cosh(angle_rad);
   double tanh_val = tanh(angle_rad);

   // Inverse hyperbolic functions
   double asinh_val = asinh(sinh_val);
   double acosh_val = acosh(cosh_val);
   double atanh_val = atanh(tanh_val);

   for(i=0;i<=100;i++)
{
   printf("-",i);

}
printf("\n");
   switch(opt)
```

```
{
case 1:
   printf("sin(%lf) = %lf\n", angle, sin_val);
   break;

case 2:
    printf("cos(%lf) = %lf\n", angle, cos_val);
    break;
case 3:
   printf("tan(%lf) = %lf\n", angle, tan_val);
   break;
case 4:
    printf("asin(%lf) = %lf\n", sin_val, asin_val);
    break;

case 5:

   printf("acos(%lf) = %lf\n", cos_val, acos_val);
   break;
case 6:
    printf("atan(%lf) = %lf\n", tan_val, atan_val);
    break;
case 7:
    printf("sinh(%lf) = %lf\n", angle, sinh_val);
    break;
case 8:
   printf("cosh(%lf) = %lf\n", angle, cosh_val);
   break;
case 9:
    printf("tanh(%lf) = %lf\n", angle, tanh_val);
    break;









}
printf("\n");
```

```
for(i=0;i<=100;i++)
{
   printf("-",i);

}



         break;

      case 3: // Exponential
         // Similar structure for exponential functions

   printf("\n option 1 exponential");
      printf("\n option 2 logarithm");
         printf("\n option 3 product of logarithms");
            printf("\n option 4 ratio of logarithms ");
             printf("\n option 5 Power of logarithms ");
              printf("\n option 6 Root of logarithms ");
               printf("\n option 7 Change of bases ");
               printf("\n");
for(i=0;i<=120;i++)
{
   printf("-");
}
printf("\n");

               printf("\n enter the value of option : ");
               scanf("%d",&opt);

   // Input values
   printf("\n Enter the value of x: ");
   scanf("%lf", &x);

   printf("\n Enter the base value: ");

   scanf("%lf", &base);

    double log_val = log(x); // Natural logarithm (base e)
   double log10_val = log10(x); // Logarithm base 10

 printf("\n");
for(i=0;i<=120;i++)
{
```

```
    printf("-");
}
printf("\n");

    switch(opt)
    {
    case 1:
        // e^x
    result = exp(x);
    printf("\ne^%.2lf = %.2lf\n", x, result);
    break;

    case 2:
         // Logarithmic functions

    printf("log(%lf) = %lf\n", x, log_val);
    printf("log10(%lf) = %lf\n", x, log10_val);

    break;

    case 3:
       // Product of logarithms
    result = log(x) * log(base);
    printf("log(%.2lf) * log(%.2lf) = %.2lf\n", x, base, result);
    break;

    case 4:
        // Ratio of logarithms
    result = log(x) / log(base);
    printf("log(%.2lf) / log(%.2lf) = %.2lf\n", x, base, result);
    break;

    case 5:
       // Power of logarithms
    result = pow(base, log(x));
    printf("%.2lf^log(%.2lf) = %.2lf\n", base, x, result);
    break;

    case 6:
         // Root of  logarithms
    result = pow(x, 1 / log(base));
    printf("%.2lf^(1/log(%.2lf)) = %.2lf\n", x, base, result);
    break;
```

```
   case 7:
        // Change of bases
   result = log(x) / log(base);
   printf("log(%.2lf) / log(%.2lf) = %.2lf (Change of bases)\n", x, base, result);
   break;









   }
   printf("\n");
    for(i=0;i<=100;i++)
   {
      printf("-");


   }



         break;

      case 4: // Algebraic

printf("\n1. Solve Linear Equation\n");
      printf("2. Solve Quadratic Equation\n");
      printf("3. Exit\n");
      printf("\n");
for(i=0;i<=120;i++)
{
   printf("-");
}
printf("\n");
      printf("Enter your choice: ");
      scanf("%d", &choice);

         // code for linear
// Function to solve linear equation ax + b = 0
void solveLinearEquation() {

   printf("Enter the coefficient 'a' in the linear equation (ax + b = 0): ");
   scanf("%lf", &a);
```

```
    printf("Enter the constant term 'b': ");
    scanf("%lf", &b);

    if (a == 0) {
        if (b == 0) {
            printf("Infinite solutions.\n");
        } else {
            printf("No solution.\n");
        }
    } else {
        double solution = -b / a;
        printf("Solution: x = %lf\n", solution);
    }
}

// Function to solve quadratic equation ax^2 + bx + c = 0
void solveQuadraticEquation() {


    printf("Enter the coefficient 'a' in the quadratic equation (ax^2 + bx + c = 0): ");
    scanf("%lf", &a);

    printf("Enter the coefficient 'b': ");
    scanf("%lf", &b);

    printf("Enter the constant term 'c': ");
    scanf("%lf", &c);

    if (a == 0) {
        printf("Not a quadratic equation.\n");
    } else {
        double discriminant = b * b - 4 * a * c;

        if (discriminant > 0) {
            double root1 = (-b + sqrt(discriminant)) / (2 * a);
            double root2 = (-b - sqrt(discriminant)) / (2 * a);
            printf("Two distinct real roots: x1 = %lf and x2 = %lf\n", root1, root2);
        } else if (discriminant == 0) {
            double root = -b / (2 * a);
            printf("One real root: x = %lf\n", root);
        } else {
            double realPart = -b / (2 * a);
```

```
        double imaginaryPart = sqrt(-discriminant) / (2 * a);
        printf("Complex roots: x1 = %lf + %lfi and x2 = %lf - %lfi\n", realPart, imaginaryPart,
realPart, imaginaryPart);
      }
   }
}

printf("\n");
for(i=0;i<=120;i++)
{
   printf("-");
}
printf("\n");


      switch (choice) {
         case 1:
            solveLinearEquation();
            break;
         case 2:
            solveQuadraticEquation();
            break;
         // Similar structure for algebraic functions
         break;}


printf("\n");
for(i=0;i<=120;i++)
{
   printf("-");
}
printf("\n");


 case 5: // Conversions
         // Similar structure for conversion functions
        {

          printf("1 length conversions");
   printf("\n 2 weight conversions");
   printf("\n 3 temperature conversions");
printf("\n");
for(i=0;i<=100;i++)
```

```
{
   printf("-");

}
printf("\n");
   printf("\nenter option : ");

   scanf("%d",&opt);

    //length code
// Function to convert between any two length units
double convertLength(double value, int fromUnit, int toUnit) {
   // Convert to meters as an intermediate step
   double meters;
   switch (fromUnit) {
      case 1: // Kilometers
         meters = value * 1000.0;
         break;
      case 2: // Meters
         meters = value;
         break;
      case 3: // Decimeters
         meters = value * 0.1;
         break;
      case 4: // Centimeters
         meters = value * 0.01;
         break;
      case 5: // Millimeters
         meters = value * 0.001;
         break;
      case 6: // Light Years
         meters = value * 9.461e15;
         break;
      case 7: // Lunar Distance
         meters = value * 384400000.0;
         break;
      case 8: // Astronomical Unit
         meters = value * 149597870700.0;
         break;
      default:
         return -1; // Invalid fromUnit
   }
```

```
    // Convert from meters to the desired toUnit
    switch (toUnit) {
        case 1: // Kilometers
            return meters / 1000.0;
        case 2: // Meters
            return meters;
        case 3: // Decimeters
            return meters * 10.0;
        case 4: // Centimeters
            return meters * 100.0;
        case 5: // Millimeters
            return meters * 1000.0;
        case 6: // Light Years
            return meters / 9.461e15;
        case 7: // Lunar Distance
            return meters / 384400000.0;
        case 8: // Astronomical Unit
            return meters / 149597870700.0;
        default:
            return -1; // Invalid toUnit
    }
}



    // weight code
   // Function to convert between any two weight units
double convertWeight(double value, int fromUnit, int toUnit) {
    // Convert to kilograms as an intermediate step
    double kilograms;
    switch (fromUnit) {
        case 1: // Kilograms
            kilograms = value;
            break;
        case 2: // Grams
            kilograms = value / 1000.0;
            break;
        case 3: // Milligrams
            kilograms = value / 1e6;
            break;
        case 4: // Tonnes
            kilograms = value * 1000.0;
            break;
```

```
      case 5: // Quintals
         kilograms = value * 100.0;
         break;
      default:
         return -1; // Invalid fromUnit
   }

   // Convert from kilograms to the desired toUnit
   switch (toUnit) {
      case 1: // Kilograms
         return kilograms;
      case 2: // Grams
         return kilograms * 1000.0;
      case 3: // Milligrams
         return kilograms * 1e6;
      case 4: // Tonnes
         return kilograms / 1000.0;
      case 5: // Quintals
         return kilograms / 100.0;
      default:
         return -1; // Invalid toUnit
   }
}

     //temperature code
   // Function to convert Celsius to Fahrenheit
double celsiusToFahrenheit(double celsius) {
   return (celsius * 9/5) + 32;
}

// Function to convert Celsius to Kelvin
double celsiusToKelvin(double celsius) {
   return celsius + 273.15;
}

// Function to convert Fahrenheit to Celsius
double fahrenheitToCelsius(double fahrenheit) {
   return (fahrenheit - 32) * 5/9;
}

// Function to convert Fahrenheit to Kelvin
double fahrenheitToKelvin(double fahrenheit) {
   return (fahrenheit - 32) * 5/9 + 273.15;
```

```
}

// Function to convert Kelvin to Celsius
double kelvinToCelsius(double kelvin) {
   return kelvin - 273.15;
}

// Function to convert Kelvin to Fahrenheit
double kelvinToFahrenheit(double kelvin) {
   return (kelvin - 273.15) * 9/5 + 32;
}

printf("\n");
for(i=0;i<=100;i++)
{
   printf("-");

}
switch(opt)
{
case 1:
     // Menu for length conversion
   printf("Select the input length unit:\n");
   printf("1. Kilometers (km)\n");
   printf("2. Meters (m)\n");
   printf("3. Decimeters (dm)\n");
   printf("4. Centimeters (cm)\n");
   printf("5. Millimeters (mm)\n");
   printf("6. Light Years (ly)\n");
   printf("7. Lunar Distance (LD)\n");
   printf("8. Astronomical Unit (AU)\n");
   printf("Enter your choice (1-8): ");
   scanf("%d", &fromUnit);

   // Input length value
   printf("\nEnter the length value: ");
   scanf("%lf", &value);

   // Menu for choosing the output length unit
   printf("\nSelect the output length unit:\n");
   printf("1. Kilometers (km)\n");
   printf("2. Meters (m)\n");
   printf("3. Decimeters (dm)\n");
```

```
    printf("4. Centimeters (cm)\n");
    printf("5. Millimeters (mm)\n");
    printf("6. Light Years (ly)\n");
    printf("7. Lunar Distance (LD)\n");
    printf("8. Astronomical Unit (AU)\n");
    printf("Enter your choice (1-8): ");
    scanf("%d", &toUnit);

    // Perform selected conversion and display result
    result = convertLength(value, fromUnit, toUnit);
    if (result >= 0) {
        printf("\nConverted length: %.6lf\n", result);
    } else {
        printf("\nInvalid input or output length unit.\n");
        return 1; // Return an error code
    }
break;
case 2:

printf("\n");
    // Menu for weight conversion
    printf("Select the input weight unit:\n");
    printf("1. Kilograms (kg)\n");
    printf("2. Grams (g)\n");
    printf("3. Milligrams (mg)\n");
    printf("4. Tonnes (t)\n");
    printf("5. Quintals (q)\n");
    printf("Enter your choice (1-5): ");
    scanf("%d", &fromUnit);

    // Input weight value
    printf("\nEnter the weight value: ");
    scanf("%lf", &value);

    // Menu for choosing the output weight unit
    printf("\nSelect the output weight unit:\n");
    printf("1. Kilograms (kg)\n");
    printf("2. Grams (g)\n");
    printf("3. Milligrams (mg)\n");
    printf("4. Tonnes (t)\n");
    printf("5. Quintals (q)\n");
    printf("Enter your choice (1-5): ");
    scanf("%d", &toUnit);
```

```
   // Perform selected conversion and display result
   result = convertWeight(value, fromUnit, toUnit);
   if (result >= 0) {
      printf("\nConverted weight: %.6lf\n", result);
   } else {
      printf("\nInvalid input or output weight unit.\n");
      return 1; // Return an error code
   }
break;

case 3:

  // Menu for temperature conversion
   printf("Select the type of temperature conversion:\n");
   printf("1. Celsius to Fahrenheit\n");
   printf("2. Celsius to Kelvin\n");
   printf("3. Fahrenheit to Celsius\n");
   printf("4. Fahrenheit to Kelvin\n");
   printf("5. Kelvin to Celsius\n");
   printf("6. Kelvin to Fahrenheit\n");
   printf("Enter your choice (1-6): ");
   scanf("%d", &choice);

   // Input temperature
   printf("Enter the temperature: ");
   scanf("%lf", &temperature);

   // Perform selected conversion and display result
   printf("\n");
for(i=0;i<=100;i++)
{
   printf("-");

}
printf("\n");
   switch (choice) {
      case 1:
         printf("%lf Celsius = %lf Fahrenheit\n", temperature,
celsiusToFahrenheit(temperature));
         break;
      case 2:
         printf("%lf Celsius = %lf Kelvin\n", temperature, celsiusToKelvin(temperature));
```

```
        break;
    case 3:
        printf("%lf Fahrenheit = %lf Celsius\n", temperature,
fahrenheitToCelsius(temperature));
         break;
    case 4:
        printf("%lf Fahrenheit = %lf Kelvin\n", temperature,
fahrenheitToKelvin(temperature));
        break;
    case 5:
        printf("%lf Kelvin = %lf Celsius\n", temperature, kelvinToCelsius(temperature));
        break;
    case 6:
        printf("%lf Kelvin = %lf Fahrenheit\n", temperature,
kelvinToFahrenheit(temperature));
        break;
    default:
        printf("Invalid choice. Please enter a number between 1 and 6.\n");
        return 1; // Return an error code
  }



printf("\n");
for(i=0;i<=100;i++)
{
  printf("-");

}
printf("\n");
break;
  } }

    case 6: // Constants

        printf("\nChoose a constant to display:\n");
    printf("1. Mass of Proton\n");
    printf("2. Mass of Neutron\n");
    printf("3. Mass of Electron\n");
    printf("4. Planck's Constant\n");
    printf("5. Atomic Mass Unit\n");
    printf("6. Avogadro's Constant\n");
    printf("7. Molar Gas Constant\n");
```

```
    printf("8. Speed of Light\n");
    printf("9. Gravity\n");
    printf("10. Atmospheric Pressure\n");
    printf("11. Exit\n");

    printf("Enter your choice: ");
    scanf("%d", &choice);
const double MASS_PROTON = 1.6726219e-27;
const double MASS_NEUTRON = 1.675e-27;
const double MASS_ELECTRON = 9.10938356e-31;

const double PLANCK_CONSTANT = 6.62607015e-34;
const double ATOMIC_MASS_UNIT = 1.66053906660e-27;

const double AVOGADROS_CONSTANT = 6.02214076e23;
const double MOLAR_GAS_CONSTANT = 8.314462618;

const double SPEED_OF_LIGHT = 299792458;
const double GRAVITY = 9.81;
const double ATMOSPHERIC_PRESSURE = 101325;
 switch (choice) {
        case 1:
           printf("Mass of Proton: %.10e kg\n", MASS_PROTON);
           break;
        case 2:
           printf("Mass of Neutron: %.10e kg\n", MASS_NEUTRON);
           break;
        case 3:
           printf("Mass of Electron: %.10e kg\n", MASS_ELECTRON);
           break;
        case 4:
           printf("Planck's Constant: %.10e J·s\n", PLANCK_CONSTANT);
           break;
        case 5:
           printf("Atomic Mass Unit: %.10e kg\n", ATOMIC_MASS_UNIT);
           break;
        case 6:
           printf("Avogadro's Constant: %.10e mol^-1\n", AVOGADROS_CONSTANT);
           break;
        case 7:
           printf("Molar Gas Constant: %.10e J·(mol·K)^-1\n", MOLAR_GAS_CONSTANT);
           break;
        case 8:
```

```
            printf("Speed of Light: %.10e m/s\n", SPEED_OF_LIGHT);
            break;
        case 9:
            printf("Gravity: %.2f m/s^2\n", GRAVITY);
            break;
        case 10:
            printf("Atmospheric Pressure: %.0f Pa\n", ATMOSPHERIC_PRESSURE);
            break;
        case 11:
            printf("Exiting the program. Goodbye!\n");
            break;
        default:
            printf("Invalid choice. Please enter a valid option.\n");
    }
        // Similar structure for constant values

        case 7:// matrix

        printf("Enter the number of rows and columns for the matrices: ");
scanf("%d %d", &rows, &cols);



// Input matrix1
printf("Enter the elements of the first matrix:\n");
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        printf("Enter element (%d, %d): ", i + 1, j + 1);
        scanf("%lf", &matrix1[i][j]);
    }
}

// Input matrix2
printf("Enter the elements of the second matrix:\n");
for (int i = 0; i < rows; ++i) {
    for (int j = 0; j < cols; ++j) {
        printf("Enter element (%d, %d): ", i + 1, j + 1);
        scanf("%lf", &matrix2[i][j]);
    }
}

int choice;
```

```
do {
    printf("\nMatrix Calculator Menu:\n");
    printf("1. Matrix Addition\n");
    printf("2. Matrix Subtraction\n");
    printf("3. Matrix Multiplication\n");
    printf("4. Matrix Trace (Matrix 1)\n");
    printf("5. Matrix Trace (Matrix 2)\n");
    printf("6. Matrix Determinant (Matrix 1)\n");
    printf("7. Matrix Determinant (Matrix 2)\n");
    printf("8. Exit\n");

    printf("Enter your choice: ");
    scanf("%d", &choice);

    switch (choice) {
        case 1:
            matrixAddition(rows, cols, matrix1, matrix2, resultm);
            displayMatrix(rows, cols, resultm);
            break;
        case 2:
            matrixSubtraction(rows, cols, matrix1, matrix2, resultm);
            displayMatrix(rows, cols, resultm);
            break;
        case 3:
            matrixMultiplication(rows, cols, rows, cols, matrix1, matrix2, resultm);
            displayMatrix(rows, cols, resultm);
            break;
        case 4:
            printf("Trace of matrix1: %.2f\n", matrixTrace(rows, cols, matrix1));
            break;
        case 5:
            printf("Trace of matrix2: %.2f\n", matrixTrace(rows, cols, matrix2));
            break;
        case 6:
            printf("Determinant of matrix1: %.2f\n", matrixDeterminant(rows, cols, matrix1));
            break;
        case 7:
            printf("Determinant of matrix2: %.2f\n", matrixDeterminant(rows, cols, matrix2));
            break;
        case 8:
            printf("Exiting the program. Goodbye!\n");
            break;
        default:
```

```
            printf("Invalid choice. Please enter a valid option.\n");
        }

    } while (choice != 8);


            break;

        default:
            printf("Invalid choice.\n");
            return 1;
    }

    printf("Result: %lf\n", resultm);
    // code for matrix functions
    // Function to display a matrix


    return 0;
}
```

6. Modules of the proposed work:

1. **Code blocks**

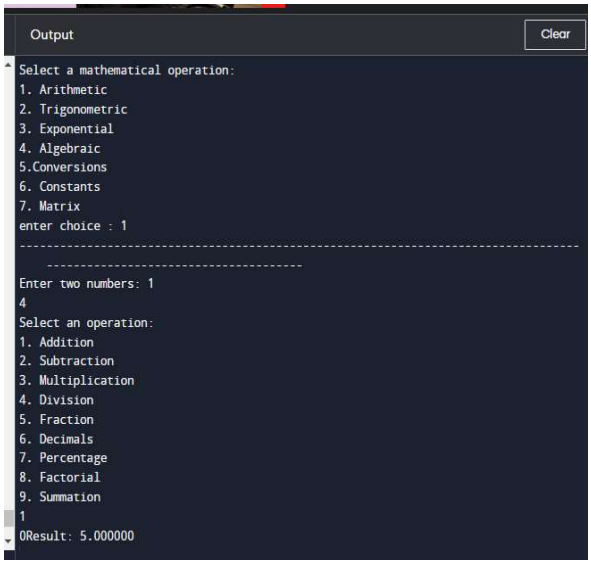
The code is done on code blocks software and the errors are simplified by online tutorials

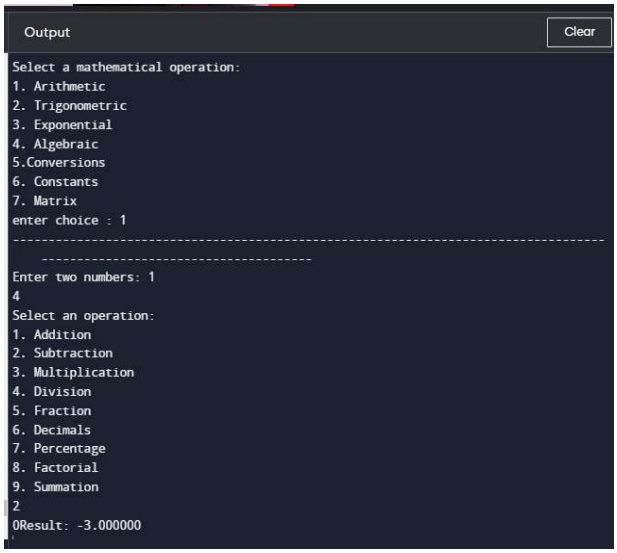
2. **Programmiz.com compiler**

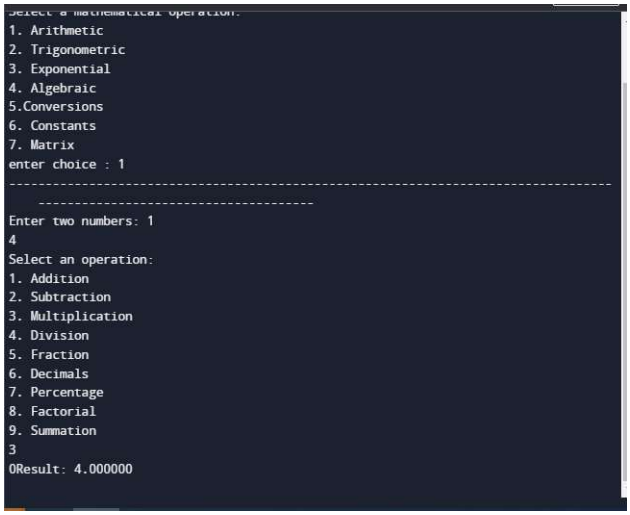
The code is complied and run on online compiler . the screenshots of the work is also taken from online compiler.

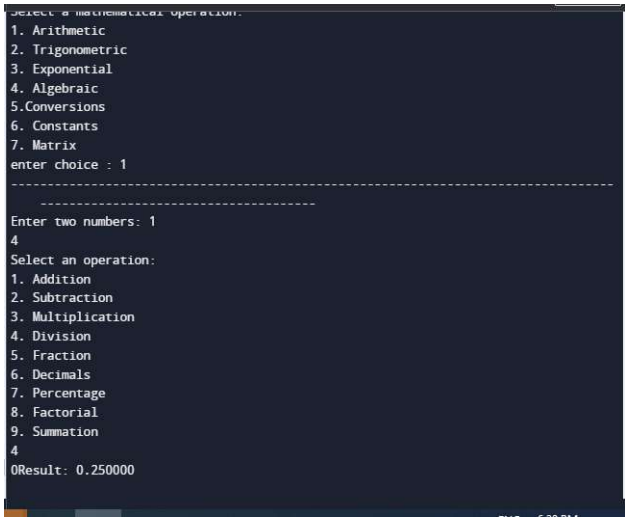
7. Results/Screenshots

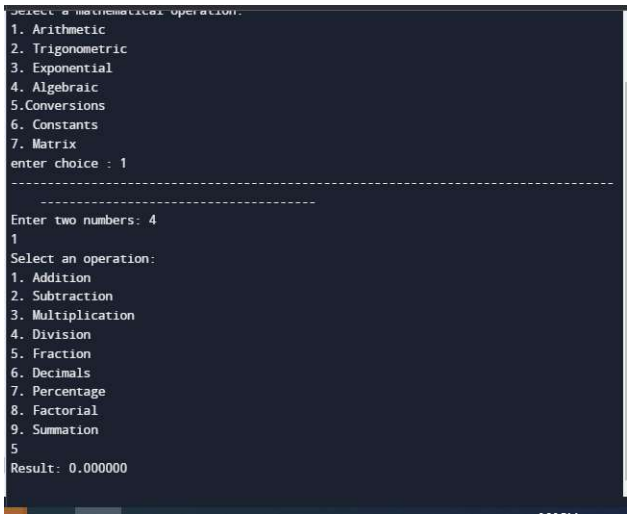
ARITHMETIC:



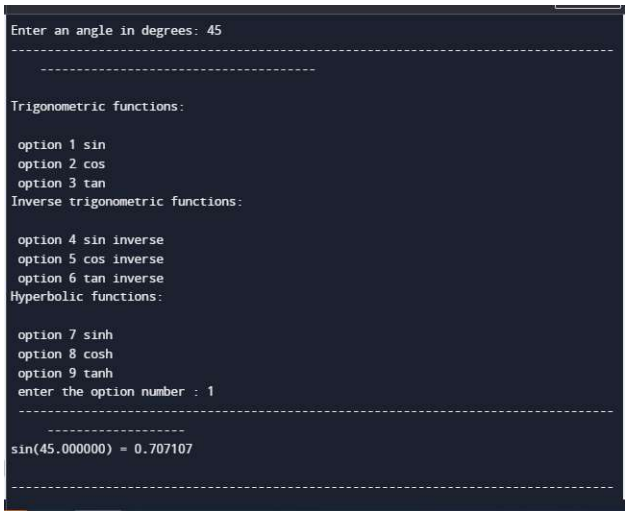


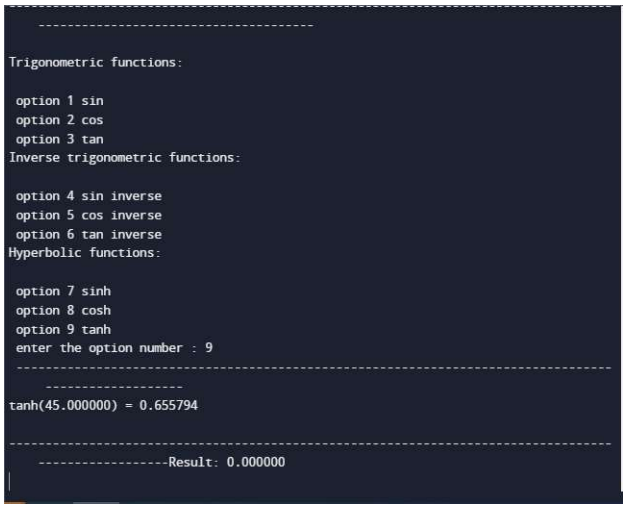

1. Arithmetic
2. Trigonometric
3. Exponential
4. Algebraic
5.Conversions
6. Constants
7. Matrix
enter choice : 1
Enter two numbers: 1
4
Select an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Fraction
6. Decimals
7. Percentage
8. Factorial
9. Summation
3
OResult: 4.000000


1. Arithmetic
2. Trigonometric
3. Exponential
4. Algebraic
5.Conversions
6. Constants
7. Matrix
enter choice : 1
Enter two numbers: 1
4
Select an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Fraction
6. Decimals
7. Percentage
8. Factorial
9. Summation
4
OResult: 0.250000


1. Arithmetic
2. Trigonometric
3. Exponential
4. Algebraic
5.Conversions
6. Constants
7. Matrix
enter choice : 1
Enter two numbers: 4
1
Select an operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Fraction
6. Decimals
7. Percentage
8. Factorial
9. Summation
5
Result: 0.000000

TRIGONOMETRIC FUNCTIONS:





EXPONENTIAL :

```
 option 1 exponential
 option 2 logarithm
 option 3 product of logarithms
 option 4 ratio of logarithms
 option 5 Power of logarithms
 option 6 Root of logarithms
 option 7 Change of bases
--------------------------------------------------------------------------------
    ------------------------------------

 enter the value of option : 1
 Enter the value of x: 4
 Enter the base value: 2
 --------------------------------------------------------------------------------
     ---------------------------------------

e^4.00 = 54.60
```

```
 option 1 exponential
 option 2 logarithm
 option 3 product of logarithms
 option 4 ratio of logarithms
 option 5 Power of logarithms
 option 6 Root of logarithms
 option 7 Change of bases
--------------------------------------------------------------------------------
    ------------------------------------

 enter the value of option : 2
 Enter the value of x: 24
 Enter the base value: 10
 --------------------------------------------------------------------------------
     ---------------------------------------
log(24.000000) = 3.178054
log10(24.000000) = 1.380211
```

```
 option 1 exponential
 option 2 logarithm
 option 3 product of logarithms
 option 4 ratio of logarithms
 option 5 Power of logarithms
 option 6 Root of logarithms
 option 7 Change of bases
--------------------------------------------------------------------------------
    ------------------------------------

 enter the value of option : 3
 Enter the value of x: 24
 Enter the base value: 10
 --------------------------------------------------------------------------------
     ---------------------------------------
log(24.00) * log(10.00) = 7.32

--------------------------------------------------------------------------------
```

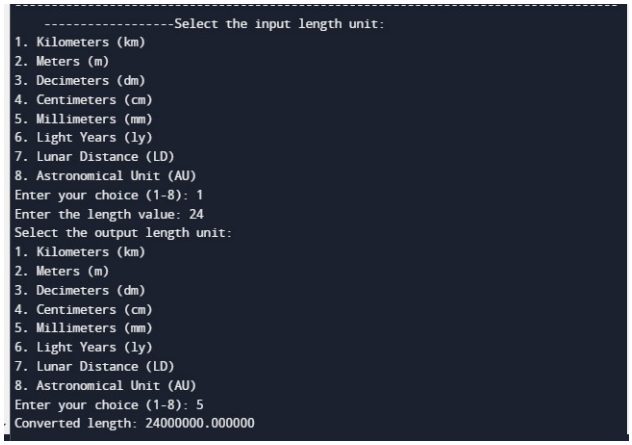
```
--------------------------------------------------------------------------------
    ------------------------------------

 option 1 exponential
 option 2 logarithm
 option 3 product of logarithms
 option 4 ratio of logarithms
 option 5 Power of logarithms
 option 6 Root of logarithms
 option 7 Change of bases
--------------------------------------------------------------------------------
    ------------------------------------

 enter the value of option : 4
 Enter the value of x: 24
 Enter the base value: 10
 --------------------------------------------------------------------------------
     ---------------------------------------
log(24.00) / log(10.00) = 1.38
```

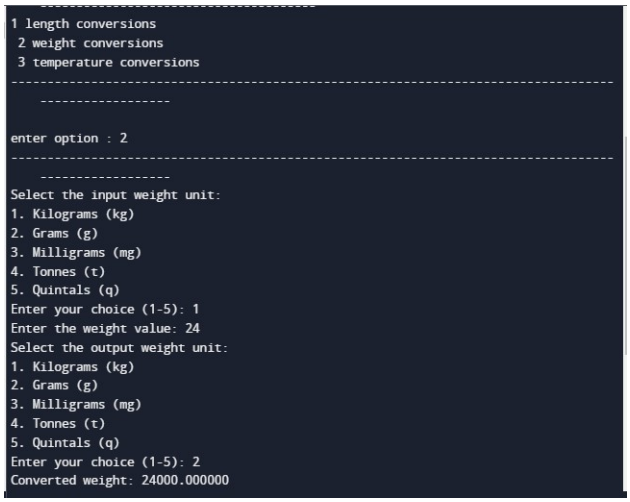
ALGEBRAIC :

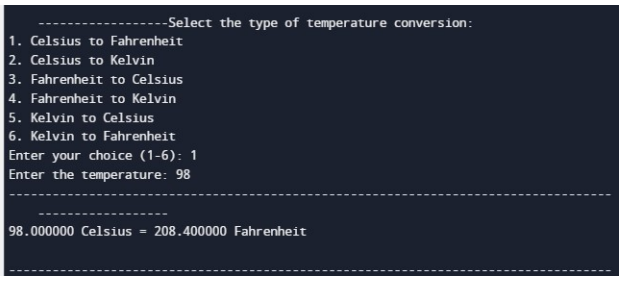
```
4

1. Solve Linear Equation
2. Solve Quadratic Equation
3. Exit

--------------------------------------------------------------------------------
    ---------------------------------------
Enter your choice: 1
--------------------------------------------------------------------------------
    ---------------------------------------
Enter the coefficient 'a' in the linear equation (ax + b = 0): 2
Enter the constant term 'b': 4
Solution: x = -2.000000

--------------------------------------------------------------------------------
```

```
    ---------------------------------------

1. Solve Linear Equation
2. Solve Quadratic Equation
3. Exit

--------------------------------------------------------------------------------
    ---------------------------------------
Enter your choice: 2
--------------------------------------------------------------------------------
    ---------------------------------------
Enter the coefficient 'a' in the quadratic equation (ax^2 + bx + c = 0): 2
Enter the coefficient 'b': 4
Enter the constant term 'c': 6
Complex roots: x1 = -1.000000 + 1.414214i and x2 = -1.000000 - 1.414214i

--------------------------------------------------------------------------------
```

CONVERSIONS:







MATRIX:

```
Enter the elements of the first matrix:
Enter element (1, 1): 1
Enter element (1, 2): 2
Enter element (2, 1): 3
Enter element (2, 2): 4
Enter the elements of the second matrix:
Enter element (1, 1): 1
Enter element (1, 2): 2
Enter element (2, 1): 3
4Enter element (2, 2):
4
Matrix Calculator Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Matrix Multiplication
4. Matrix Trace (Matrix 1)
5. Matrix Trace (Matrix 2)
6. Matrix Determinant (Matrix 1)
7. Matrix Determinant (Matrix 2)
8. Exit
Enter your choice: 1
Matrix:
6.00    8.00
0.00    0.00
```

```
Enter the number of rows and columns for the matrices: 2
2
Enter the elements of the first matrix:
Enter element (1, 1): 1
Enter element (1, 2): 2
Enter element (2, 1): 3
Enter element (2, 2): 4
Enter the elements of the second matrix:
Enter element (1, 1): 1
Enter element (1, 2): 2
Enter element (2, 1): 3
Enter element (2, 2): 4
Matrix Calculator Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Matrix Multiplication
4. Matrix Trace (Matrix 1)
5. Matrix Trace (Matrix 2)
6. Matrix Determinant (Matrix 1)
7. Matrix Determinant (Matrix 2)
8. Exit
Enter your choice: 2
Matrix:
0.00    0.00
0.00    0.00
```

```
    ------------------------------------
Enter the number of rows and columns for the matrices: 2
2
Enter the elements of the first matrix:
Enter element (1, 1): 1
Enter element (1, 2): 2
Enter element (2, 1): 3
Enter element (2, 2): 4
Enter the elements of the second matrix:
Enter element (1, 1): 1
Enter element (1, 2): 2
Enter element (2, 1): 3
Enter element (2, 2): 4
Matrix Calculator Menu:
1. Matrix Addition
2. Matrix Subtraction
3. Matrix Multiplication
4. Matrix Trace (Matrix 1)
5. Matrix Trace (Matrix 2)
6. Matrix Determinant (Matrix 1)
7. Matrix Determinant (Matrix 2)
8. Exit
Enter your choice: 4
Trace of matrix1: 3.00
```

CONSTANTS:

```
--------------------------------------

Choose a constant to display:
1. Mass of Proton
2. Mass of Neutron
3. Mass of Electron
4. Planck's Constant
5. Atomic Mass Unit
6. Avogadro's Constant
7. Molar Gas Constant
8. Speed of Light
9. Gravity
10. Atmospheric Pressure
11. Exit
Enter your choice: 1
Mass of Proton: 1.6726219000e-27 kg
Enter the number of rows and columns for the matrices:
```

```
--------------------------------------

Choose a constant to display:
1. Mass of Proton
2. Mass of Neutron
3. Mass of Electron
4. Planck's Constant
5. Atomic Mass Unit
6. Avogadro's Constant
7. Molar Gas Constant
8. Speed of Light
9. Gravity
10. Atmospheric Pressure
11. Exit
Enter your choice: 2
Mass of Neutron: 1.6750000000e-27 kg
```

```
--------------------------------------

Choose a constant to display:
1. Mass of Proton
2. Mass of Neutron
3. Mass of Electron
4. Planck's Constant
5. Atomic Mass Unit
6. Avogadro's Constant
7. Molar Gas Constant
8. Speed of Light
9. Gravity
10. Atmospheric Pressure
11. Exit
Enter your choice: 3
Mass of Electron: 9.1093835600e-31 kg
Enter the number of rows and columns for the matrices:
```

```
Choose a constant to display:
1. Mass of Proton
2. Mass of Neutron
3. Mass of Electron
4. Planck's Constant
5. Atomic Mass Unit
6. Avogadro's Constant
7. Molar Gas Constant
8. Speed of Light
9. Gravity
10. Atmospheric Pressure
11. Exit
Enter your choice: 4
Planck's Constant: 6.6260701500e-34 J·s
Enter the number of rows and columns for the matrices:
```

8. Conclusion:

The main purpose of our project is to minimize the cost of scientific calculator

It is hard to buy calculator in such prize so this program is introduced for such needs.

In conclusion, the development of a scientific calculator mini-project in the C programming language has provided valuable insights into various programming concepts and practical applications. The project successfully implemented essential features such as arithmetic operations, trigonometric functions, logarithmic functions, and more, showcasing the versatility of C in handling complex mathematical computations.

Throughout the development process, key programming principles such as modularization, code organization, and efficient algorithm design were employed to ensure a clean and maintainable codebase. The utilization of functions, control structures, and data structures enhanced the overall readability and scalability of the project.

Moreover, this mini-project served as an excellent opportunity to apply error handling mechanisms, ensuring the calculator's robustness by preventing unintended behavior and providing meaningful feedback to users in case of invalid inputs or operations.

By implementing the scientific calculator, a foundational understanding of both C programming and mathematical concepts was reinforced. The project also laid the groundwork for further exploration and expansion, allowing for the incorporation of additional features, user interfaces, or optimizations in future iterations.

In summary, the scientific calculator mini-project not only fulfilled its primary objective of providing a functional tool for mathematical computations but also contributed to the development of programming skills and a deeper appreciation for the intersection of mathematics and computer science. This experience serves as a stepping stone for continued learning and exploration within the realm of software development and algorithmic problem-solving.

9. References:

When creating a scientific calculator mini-project in C language, you may not have formal references in the same way you would for a research paper. However, you can draw inspiration from various sources, including documentation, tutorials, and books on C programming and mathematical algorithms. Here are some general references that may be helpful:

1. **C Programming Language Documentation:**
   - C Standard Library - The official documentation for the C Standard Library, which includes functions like **sin()**, **cos()**, **log()**, etc.
2. **Math Library Documentation:**
   - Math Library Functions in C - Detailed documentation for mathematical functions in C, which can be useful for implementing calculator functions.
3. **C Programming Books:**
   - "C Programming Absolute Beginner's Guide" by Perry and Miller.
   - "C Programming for the Absolute Beginner" by Vine.
   - "C Programming for the Absolute Beginner, Second Edition" by Vine and Vine.
4. **Online Tutorials and Resources:**
   - GeeksforGeeks (C Language Section): GeeksforGeeks C Programming
   - Tutorialspoint (C Programming): Tutorialspoint C Programming
5. **Open Source Projects:**
   - Explore open-source calculator projects on platforms like GitHub for implementation ideas and coding practices.