PRACTICE C++

160

solved

EXERCISES

to accelerate your learning

RUHAN CONCEIÇÃO

PRAC-

TICE C++

160 Solved Exercises

to Accelerate your

Learning

Ruhan Avila da Conceição

Preface

Welcome to this book where solved and commented C++ exercises are presented. In this book, you will find a collection of over 160 exercises designed to help you improve your programming skills in this powerful language.

Learning to program involves not only understanding theoretical concepts but also applying those concepts in real-life situations. That's exactly what you will find in this book: a wide variety of problems ranging from basic fundamentals to more complex challenges.

As you progress through the exercises, you will be challenged with tasks involving mathematical formula manipulation, strings, condi-

tionals, loops, vector manipulation, matrices, and much more.

The main goal of this book is to provide a practical and comprehensive resource for programmers seeking improvement. Whether you are a beginner in C++ looking to solidify your knowledge or an experienced programmer wishing to deepen your expertise, these exercises will serve as an excellent study guide and reference. This book is also suitable for teachers who would like to have a rich collection of solved Programming Logic exercises to create exercises and questions for their students.

Enjoy this learning journey and dive into the solved and commented C++ exercises. Prepare yourself for stimulating challenges, creative solutions, and a unique opportunity to enhance your programming skills.

This book was written using artificial intelligence tools in content creation, but all materials have been reviewed and edited by the author to deliver a final high-quality product.

Happy reading, happy studying, and have fun exploring the fascinating world of C++ programming

Ruhan Avila da Conceição.

Summary

Introduction

If you have acquired this book, you want to start programming and be logically challenged as soon as possible, without wanting to read a sermon on the mount. But it is important to highlight a few things before we begin.

Even though many exercises may be considered easy, if you are new to this programming journey, it is important for you to first try to solve the problem on your own before looking at the solution. There is more than one possible solution to the same problem, and you need to think and develop your own solution. Then, you can compare it with the proposed one in the book, identify the strengths of each, and try to learn a little more.

If the exercise is too difficult and you can't solve it, move on to the next one and try again the next day. Don't immediately jump to the answer, even if you can't solve it, and definitely don't look at the answer without even attempting to solve it.

Learning programming logic is not about getting the answer; it's about the journey you take to arrive at the answer.

With that being said, the remaining chapters of this book are divided according to the programming topics covered in the proposed exercises.

From now on, it's all up to you!

Basic Exercises

1. Write a program that prints "Hello World!" on the screen.

```
#include <iostream>

int main() {
   std::cout << "Hello World!" << std::endl;
   return 0;
}
```

2. Write a program that prints "Hello" on one line and "World!" on the bottom line, using two different commands.

```
#include <iostream>

int main() {
```

```
    std::cout << "Hello" << std::endl;
    std::cout << "World!" << std::endl;
    return 0;
}
```

3. Write a program that prints "Hello" on one line and "World!" on the bottom line, using only one command to print.

```
#include <iostream>

int main() {
    std::cout << "Hello" << std::endl << "World!" <<
std::endl;
    return 0;
}
```

4. Write a program that prints "Hello" on one line and "World!" on the bottom line, leaving a blank line between the words.

```
#include <iostream>

int main() {
   std::cout << "Hello" << std::endl << std::endl <<
"World!" << std::endl;
   return 0;
}
```

5. Write a program that prints "Hello World!" (in the same line) on the screen, using one command to print "Hello" and another command to print "World!".

```
#include <iostream>

int main() {
   std::cout << "Hello ";
```

```
    std::cout << "World!" << std::endl;
    return 0;
}
```

6. Write a program that prints "Hello World!" (in the same line) on the screen, using one command to print "Hello", another command to print "World", and another to print "!".

```
#include <iostream>

int main() {
    std::cout << "Hello";
    std::cout << " World";
    std::cout << "!" << std::endl;
    return 0;
}
```

7. Make a program that asks for the user's name, and prints the name on the screen immediately after.

```
#include <iostream>
#include <string>

int main() {
   std::string name;

    std::cout << "Please enter your name: ";
   std::getline(std::cin, name);
   std::cout << "Hello, " << name << "!" << std::endl;

    return 0;
}
```

8. Make a program that reads the names of three people in sequence, followed by their respective ages, and prints each name with their respective ages on the screen.

```
#include <iostream>
#include <string>

int main() {
   std::string name1, name2, name3;
   int age1, age2, age3;

    // Read details for the first person
   std::cout << "Enter the name of the first person: ";
   std::getline(std::cin, name1);
   std::cout << "Enter the age of " << name1 << ": ";
   std::cin >> age1;
   std::cin.ignore();  // To clear the newline left by
the previous input
```

```
    // Read details for the second person
    std::cout << "Enter the name of the second person: ";
    std::getline(std::cin, name2);
    std::cout << "Enter the age of " << name2 << ": ";
    std::cin >> age2;
    std::cin.ignore();  // To clear the newline left by the previous input

    // Read details for the third person
    std::cout << "Enter the name of the third person: ";
    std::getline(std::cin, name3);
    std::cout << "Enter the age of " << name3 << ": ";
    std::cin >> age3;
    std::cin.ignore();  // To clear the newline left by the previous input

    // Output the details
    std::cout << std::endl;
    std::cout << name1 << " is " << age1 << " years old." << std::endl;
```

```
    std::cout << name2 << " is " << age2 << " years
old." << std::endl;
    std::cout << name3 << " is " << age3 << " years
old." << std::endl;

    return 0;
}
```

9. Make a program that reads the names of three people in sequence, followed by their respective ages and heights, and prints the data in table format on the screen.

```
#include <iostream>
#include <iomanip>
#include <string>

int main() {
    std::string name1, name2, name3;
    int age1, age2, age3;
    float height1, height2, height3;
```

```
    // Read details for the first person
    std::cout << "Enter the name of the first person: ";
    std::getline(std::cin, name1);
    std::cout << "Enter the age of " << name1 << ": ";
    std::cin >> age1;
    std::cout << "Enter the height (in meters) of " <<
name1 << ": ";
    std::cin >> height1;
    std::cin.ignore();  // To clear the newline left by
the previous input

    // Read details for the second person
    std::cout << "Enter the name of the second per-
son: ";
    std::getline(std::cin, name2);
    std::cout << "Enter the age of " << name2 << ": ";
    std::cin >> age2;
    std::cout << "Enter the height (in meters) of " <<
name2 << ": ";
    std::cin >> height2;
    std::cin.ignore();  // To clear the newline left by
the previous input
```

```
    // Read details for the third person
    std::cout << "Enter the name of the third person:
";
    std::getline(std::cin, name3);
    std::cout << "Enter the age of " << name3 << ": ";
    std::cin >> age3;
    std::cout << "Enter the height (in meters) of " <<
name3 << ": ";
    std::cin >> height3;
    std::cin.ignore();  // To clear the newline left by
the previous input

    // Output the details in table format
    std::cout << std::endl;
    std::cout << std::left << std::setw(20) << "Name"
              << std::setw(5) << "Age"
              << std::setw(10) << "Height(m)" << std::endl;
    std::cout << "-----------------------------------" <<
std::endl;
    std::cout << std::left << std::setw(20) << name1
              << std::setw(5) << age1
              << std::setw(10) << height1 << std::endl;
```

```
    std::cout << std::left << std::setw(20) << name2
              << std::setw(5) << age2
              << std::setw(10) << height2 << std::endl;
    std::cout << std::left << std::setw(20) << name3
              << std::setw(5) << age3
              << std::setw(10) << height3 << std::endl;

    return 0;
}
```

10. Make a program that reads the names of three people in sequence, followed by reading their respective ages and heights, and prints the data on the screen in table format, using dashes (-) to separate rows and pipes (|) to separate columns.

```
#include <iostream>
#include <iomanip>
#include <string>
```

```
int main() {
   std::string name1, name2, name3;
   int age1, age2, age3;
   float height1, height2, height3;

    // Read details for the first person
   std::cout << "Enter the name of the first person: ";
   std::getline(std::cin, name1);
   std::cout << "Enter the age of " << name1 << ": ";
   std::cin >> age1;
   std::cout << "Enter the height (in meters) of " <<
name1 << ": ";
   std::cin >> height1;
   std::cin.ignore();  // To clear the newline left by
the previous input

    // Read details for the second person
   std::cout << "Enter the name of the second per-
son: ";
   std::getline(std::cin, name2);
   std::cout << "Enter the age of " << name2 << ": ";
   std::cin >> age2;
```

```
    std::cout << "Enter the height (in meters) of " <<
name2 << ": ";
    std::cin >> height2;
    std::cin.ignore();  // To clear the newline left by
the previous input

     // Read details for the third person
    std::cout << "Enter the name of the third person:
";
    std::getline(std::cin, name3);
    std::cout << "Enter the age of " << name3 << ": ";
    std::cin >> age3;
    std::cout << "Enter the height (in meters) of " <<
name3 << ": ";
    std::cin >> height3;
    std::cin.ignore();  // To clear the newline left by
the previous input

     // Output the details in table format
    std::cout << std::endl;
    std::cout << std::left << std::setw(20) << "| Name"
              << std::setw(5) << "| Age"
              << std::setw(10) << "| Height(m) |" <<
```

```
std::endl;
  std::cout << "|------------------|-----|---------|" <<
std::endl;
  std::cout << std::left << std::setw(20) << "| " +
name1
        << std::setw(5) << "| " + std::to_string(age1)
        << std::setw(10) << "| " + std::
to_string(height1) + " |" << std::endl;
  std::cout << "|------------------|-----|---------|" <<
std::endl;
  std::cout << std::left << std::setw(20) << "| " +
name2
        << std::setw(5) << "| " + std::to_string(age2)
        << std::setw(10) << "| " + std::
to_string(height2) + " |" << std::endl;
  std::cout << "|------------------|-----|---------|" <<
std::endl;
  std::cout << std::left << std::setw(20) << "| " +
name3
        << std::setw(5) << "| " + std::to_string(age3)
        << std::setw(10) << "| " + std::
to_string(height3) + " |" << std::endl;
```

```
    std::cout << "|--------------------|-----|----------|" <<
std::endl;

    return 0;
}
```

Mathematical Formulas

11. Write a program that prompts the user for two numbers and displays the addition, subtraction, multiplication, and division between them.

```
#include <iostream>

int main() {
   double num1, num2;

    // Prompt the user for two numbers
   std::cout << "Enter the first number: ";
   std::cin >> num1;

    std::cout << "Enter the second number: ";
   std::cin >> num2;
```

```
    // Display the results
   std::cout << "Addition: " << num1 + num2 <<
std::endl;
   std::cout << "Subtraction: " << num1 - num2 <<
std::endl;
   std::cout << "Multiplication: " << num1 * num2 <<
std::endl;
   std::cout << "Division: " << num1 / num2 <<
std::endl;

    return 0;
}
```

12. Write a program that calculates the arithmetic mean of two numbers entered by the user.

```
#include <iostream>

int main() {
   double num1, num2;
```

```
    // Prompt the user for two numbers
    std::cout << "Enter the first number: ";
    std::cin >> num1;

    std::cout << "Enter the second number: ";
    std::cin >> num2;

    // Calculate the arithmetic mean
    double mean = (num1 + num2) / 2;

    // Display the result
    std::cout << "The arithmetic mean of " << num1
<< " and " << num2 << " is: " << mean << std::endl;

    return 0;
}
```

13. Create a program that calculates and displays the arithmetic mean of three numbers entered by the user.

```
#include <iostream>
```

```
int main() {
   double num1, num2, num3;

    // Prompt the user for three numbers
   std::cout << "Enter the first number: ";
   std::cin >> num1;

    std::cout << "Enter the second number: ";
   std::cin >> num2;

    std::cout << "Enter the third number: ";
   std::cin >> num3;

    // Calculate the arithmetic mean
   double mean = (num1 + num2 + num3) / 3;

    // Display the result
   std::cout << "The arithmetic mean of " << num1
<< ", " << num2 << " and " << num3 << " is: " <<
mean << std::endl;

    return 0;
}
```

14. Write a program that reads two numbers, the first being the base and the second the exponent, and then printing the first number raised to the second.

```
#include <iostream>
#include <cmath>  // Required for the pow() function

int main() {
   double base, exponent;

    // Prompt the user for the base and the exponent
   std::cout << "Enter the base: ";
   std::cin >> base;

    std::cout << "Enter the exponent: ";
   std::cin >> exponent;

    // Calculate the power using the pow() function
   double result = std::pow(base, exponent);
```

```
    // Display the result
   std::cout << base << " raised to the power of " <<
exponent << " is: " << result << std::endl;

    return 0;
}
```

15. Write a program that reads a number and prints the square root of the number on the screen.

```
#include <iostream>
#include <cmath>  // Required for the sqrt()
function

int main() {
   double number;

    // Prompt the user for a number
   std::cout << "Enter a number: ";
   std::cin >> number;
```

```
    // Ensure the number is non-negative
    if (number < 0) {
        std::cout << "The number is negative, and its square root is not real." << std::endl;
        return 1;
    }

    // Calculate the square root using the sqrt() function
    double sqrt_value = std::sqrt(number);

    // Display the result
    std::cout << "The square root of " << number << " is: " << sqrt_value << std::endl;

    return 0;
}
```

16. Write a program that calculates the geometric mean of three numbers entered by the user → $\text{GeoMean} = \sqrt[3]{n_1 \times n_2 \times n_3}$

```
#include <iostream>
#include <cmath>  // Required for the cbrt() and pow() functions

int main() {
   double num1, num2, num3;

    // Prompt the user for three numbers
   std::cout << "Enter the first number: ";
   std::cin >> num1;

    std::cout << "Enter the second number: ";
   std::cin >> num2;

    std::cout << "Enter the third number: ";
   std::cin >> num3;
```

```
    // Calculate the geometric mean
    double product = num1 * num2 * num3;
    double geometric_mean = std::cbrt(product);

    // Alternatively, you can use the pow() function:
    // double geometric_mean = std::pow(product,
1.0 / 3.0);

    // Display the result
    std::cout << "The geometric mean of " << num1
<< ", " << num2 << ", and " << num3 << " is: " << geo-
metric_mean << std::endl;

    return 0;
}
```

17. Write a program that calculates the BMI of an individual, using the formula BMI = weight / height²

```
#include <iostream>
#include <cmath>  // Required for the pow()
```

function

```
int main() {
   double weight, height;

    // Prompt the user for weight and height
   std::cout << "Enter your weight in kilograms: ";
   std::cin >> weight;

    std::cout << "Enter your height in meters: ";
   std::cin >> height;

    // Calculate the BMI
   double bmi = weight / std::pow(height, 2);

    // Display the BMI
   std::cout << "Your BMI is: " << bmi << std::endl;

    return 0;
}
```

18. Create a program that calculates and displays the perimeter of a circle, prompting the user for the radius, using the formula P = 2πr

```
#include <iostream>
#include <cmath>  // Required for the M_PI
constant

int main() {
   double radius;

    // Prompt the user for the radius
   std::cout << "Enter the radius of the circle: ";
   std::cin >> radius;

    // Calculate the perimeter/circumference
   double perimeter = 2 * M_PI * radius;

    // Display the result
   std::cout << "The perimeter (circumference)
of the circle with radius " << radius << " is: " <<
```

```
perimeter << std::endl;

    return 0;
}
```

19. Write a program that calculates the area of a circle from the radius, using the formula $A = \pi r^2$

```
#include <iostream>
#include <cmath>  // Required for the M_PI
constant

int main() {
   double radius;

    // Prompt the user for the radius
   std::cout << "Enter the radius of the circle: ";
   std::cin >> radius;

    // Calculate the area
   double area = M_PI * std::pow(radius, 2);
```

```
    // Display the result
   std::cout << "The area of the circle with radius "
<< radius << " is: " << area << std::endl;

    return 0;
}
```

20. Write a program that calculates the delta of a quadratic equation ($\Delta = b^2 - 4ac$).

```
#include <iostream>
#include <cmath>

int main() {
   double a, b, c;

    // Prompt the user for the coefficients of the
quadratic equation
   std::cout << "Enter the coefficient a: ";
   std::cin >> a;

    std::cout << "Enter the coefficient b: ";
   std::cin >> b;
```

```
    std::cout << "Enter the coefficient c: ";
    std::cin >> c;

    // Calculate delta (Δ)
    double delta = std::pow(b, 2) - 4 * a * c;

    // Display the result
    std::cout << "The delta (Δ) of the equation is: " <<
delta << std::endl;

    return 0;
}
```

21. Write a program that calculates the perimeter and area of a rectangle, using the formulas P = 2(w + l) and A = wl, where w is the width and l is the length

```
#include <iostream>

int main() {
    double width, length;
```

```
    // Prompt the user for the width and length of
the rectangle
   std::cout << "Enter the width of the rectangle: ";
   std::cin >> width;

    std::cout << "Enter the length of the rectangle: ";
   std::cin >> length;

    // Calculate the perimeter and area
   double perimeter = 2 * (width + length);
   double area = width * length;

    // Display the results
   std::cout << "The perimeter of the rectangle is: "
<< perimeter << std::endl;
   std::cout << "The area of the rectangle is: " << area
<< std::endl;

    return 0;
}
```

22. Write a program that calculates the perimeter and area of a triangle, using the formulas P = a + b + c and A = (b * h) / 2, where a, b and c are the sides of the triangle and h is the height relative to the side B.

```
#include <iostream>

int main() {
  double a, b, c, h;

   // Prompt the user for the sides and height of the triangle
  std::cout << "Enter the length of side a: ";
  std::cin >> a;

   std::cout << "Enter the length of side b: ";
  std::cin >> b;

   std::cout << "Enter the length of side c: ";
```

```
    std::cin >> c;

     std::cout << "Enter the height relative to side b: ";
    std::cin >> h;

     // Calculate the perimeter and area
    double perimeter = a + b + c;
    double area = (b * h) / 2;

     // Display the results
    std::cout << "The perimeter of the triangle is: " <<
perimeter << std::endl;
    std::cout << "The area of the triangle is: " << area
<< std::endl;

     return 0;
}
```

23. Write a program that calculates the average velocity of an object, using the formula v = Δs/Δt, where v is the average velocity, Δs is the space variation, and Δt is the time variation

```
#include <iostream>

int main() {
   double delta_s, delta_t;

    // Prompt the user for the space variation and time variation
   std::cout << "Enter the space variation (Δs) in meters: ";
   std::cin >> delta_s;

    std::cout << "Enter the time variation (Δt) in seconds: ";
   std::cin >> delta_t;

    // Calculate the average velocity
```

```
    double average_velocity = delta_s / delta_t;

     // Display the result
    std::cout << "The average velocity (v) is: " << aver-
age_velocity << " m/s" << std::endl;

     return 0;
}
```

24. Write a program that calculates the kinetic energy of a moving object, using the formula E = (mv^2) / 2, where E is the kinetic energy, m is the mass of the object, and v is the velocity.

```
#include <iostream>
#include <cmath>  // Required for the pow()
function

int main() {
    double mass, velocity;
```

```
    // Prompt the user for the mass and velocity of the object
    std::cout << "Enter the mass of the object (in kg): ";
    std::cin >> mass;

    std::cout << "Enter the velocity of the object (in m/s): ";
    std::cin >> velocity;

    // Calculate the kinetic energy
    double kinetic_energy = (mass * std::pow(velocity, 2)) / 2;

    // Display the result
    std::cout << "The kinetic energy of the object is: " << kinetic_energy << " Joules" << std::endl;

    return 0;
}
```

25. Write a program that calculates the work done by a force acting on an object, using the formula W = F * d, where W is the work, F is the applied force, and d is the distance traveled by the object.

```
#include <iostream>

int main() {
   double force, distance;

    // Prompt the user for the applied force and distance
   std::cout << "Enter the force applied on the object (in Newtons): ";
   std::cin >> force;

    std::cout << "Enter the distance traveled by the object (in meters): ";
   std::cin >> distance;

    // Calculate the work done
```

```
    double work = force * distance;

     // Display the result
    std::cout << "The work done by the force is: " <<
work << " Joules" << std::endl;

     return 0;
}
```

26. Write a program that calculates the n-th term of an arithmetic progression given the value of the first term, the common difference (ratio), and the value of n read from the user →

$$a_n = a_1 + (n - 1) \times r$$

```
#include <iostream>

int main() {
    double a1, d;
    int n;
```

```
    // Prompt the user for the first term, common difference, and n
    std::cout << "Enter the first term of the arithmetic progression: ";
    std::cin >> a1;

    std::cout << "Enter the common difference of the arithmetic progression: ";
    std::cin >> d;

    std::cout << "Enter the term number (n) you want to calculate: ";
    std::cin >> n;

    // Calculate the n-th term
    double an = a1 + (n-1) * d;

    // Display the result
    std::cout << "The " << n << "-th term of the arithmetic progression is: " << an << std::endl;

    return 0;
}
```

27. Write a program that reads the *x* and *y* position of two points in the Cartesian plane, and calculates the distance between them

```
#include <iostream>
#include <cmath>  // For the sqrt() and pow() functions

int main() {
   double x1, y1, x2, y2;

    // Prompt the user for the coordinates of the first point
   std::cout << "Enter the x and y coordinates of the first point (separated by space): ";
   std::cin >> x1 >> y1;

    // Prompt the user for the coordinates of the second point
   std::cout << "Enter the x and y coordinates of the second point (separated by space): ";
```

```
    std::cin >> x2 >> y2;

    // Calculate the distance between the two
points
    double distance = sqrt(pow(x2 - x1, 2) + pow(y2 -
y1, 2));

    // Display the result
    std::cout << "The distance between the two
points is: " << distance << std::endl;

    return 0;
}
```

28. Create a program that prompts the user for the radius of a sphere and calculates and displays its volume.

```
#include <iostream>
#include <cmath>  // For the pow() function

const double PI = 3.14;  // Value of Pi
```

```
int main() {
    double radius;

    // Prompt the user for the radius of the sphere
    std::cout << "Enter the radius of the sphere: ";
    std::cin >> radius;

    // Calculate the volume of the sphere
    double volume = (4.0 / 3.0) * PI * pow(radius, 3);

    // Display the result
    std::cout << "The volume of the sphere with radius " << radius << " is: " << volume << " cubic units"
<< std::endl;

    return 0;
}
```

29. Make a program that reads the resistance of two resistors and displays the resulting resistance value when connected in series, adding the values, and in parallel using the formula (R1 * R2) / (R1 + R2)

```
#include <iostream>

int main() {
   double R1, R2;

    // Prompt the user for the resistances of the two
resistors
   std::cout << "Enter the resistance of the first re-
sistor (in ohms): ";
   std::cin >> R1;

    std::cout << "Enter the resistance of the second
resistor (in ohms): ";
   std::cin >> R2;
```

```
    // Calculate the resulting resistance when the resistors are connected in series
    double series_resistance = R1 + R2;

    // Calculate the resulting resistance when the resistors are connected in parallel
    double parallel_resistance = (R1 * R2) / (R1 + R2);

    // Display the results
    std::cout << "Resulting resistance when connected in series: " << series_resistance << " ohms" << std::endl;
    std::cout << "Resulting resistance when connected in parallel: " << parallel_resistance << " ohms" << std::endl;

    return 0;
}
```

30. Make a program that reads a value of a product and the tax (in percentage), and calculates the final value of it.

```
#include <iostream>

int main() {
  double productValue, taxRate;

   // Prompt the user for the product value
  std::cout << "Enter the value of the product: ";
  std::cin >> productValue;

   // Prompt the user for the tax rate (in percentage)
  std::cout << "Enter the tax rate (in percentage): ";
  std::cin >> taxRate;

   // Calculate the tax amount
  double taxAmount = (taxRate / 100) * productValue;

   // Calculate the final product value including
```

```
the tax
    double finalValue = productValue + taxAmount;

     // Display the results
    std::cout << "Final value of the product after ap-
plying the tax: " << finalValue << std::endl;

     return 0;
}
```

Conditionals

31. Make a program that asks for a person's age and displays whether they are of legal age or not (age >= 18).

```
#include <iostream>

int main() {
  int age;

   // Prompt the user for their age
  std::cout << "Please enter your age: ";
  std::cin >> age;

   // Check if the person is of legal age or not
  if (age >= 18) {
    std::cout << "You are of legal age." << std::endl;
  } else {
    std::cout << "You are not of legal age." <<
std::endl;
```

```
    }

    return 0;
}
```

32. Write a program that reads a number and reports whether it is odd or even.

```
#include <iostream>

int main() {
    int number;

    // Prompt the user for a number
    std::cout << "Please enter a number: ";
    std::cin >> number;

    // Determine if the number is odd or even
    if (number % 2 == 0) {
        std::cout << "The number is even." << std::endl;
    } else {
        std::cout << "The number is odd." << std::endl;
    }
```

```
    return 0;
}
```

33. Write a program that reads a number and reports whether it is positive, negative or zero.

```
#include <iostream>

int main() {
   double number;

    // Prompt the user for a number
   std::cout << "Please enter a number: ";
   std::cin >> number;

    // Determine if the number is positive, negative,
or zero
   if (number > 0) {
      std::cout << "The number is positive." <<
std::endl;
   } else if (number < 0) {
```

```
    std::cout << "The number is negative." <<
std::endl;
  } else {
    std::cout << "The number is zero." << std::endl;
  }

   return 0;
}
```

34. Create a program that reads three numbers and checks if their sum is positive, negative or equal to zero

```
#include <iostream>

int main() {
  double num1, num2, num3;

   // Prompt the user for three numbers
  std::cout << "Enter the first number: ";
  std::cin >> num1;

   std::cout << "Enter the second number: ";
```

```
    std::cin >> num2;

    std::cout << "Enter the third number: ";
    std::cin >> num3;

    // Calculate the sum of the three numbers
    double sum = num1 + num2 + num3;

    // Determine if the sum is positive, negative, or
zero
    if (sum > 0) {
        std::cout << "The sum is positive." << std::endl;
    } else if (sum < 0) {
        std::cout << "The sum is negative." << std::endl;
    } else {
        std::cout << "The sum is zero." << std::endl;
    }

    return 0;
}
```

35. Write a program that reads two numbers and tells you which one is bigger.

```
#include <iostream>

int main() {
   double num1, num2;

    // Prompt the user for two numbers
   std::cout << "Enter the first number: ";
   std::cin >> num1;

    std::cout << "Enter the second number: ";
   std::cin >> num2;

    // Determine and display which number is
bigger
   if (num1 > num2) {
      std::cout << "The first number (" << num1 << ")
is bigger." << std::endl;
   } else if (num1 < num2) {
      std::cout << "The second number (" << num2 <<
") is bigger." << std::endl;
```

```
    } else {
        std::cout << "Both numbers are equal." <<
std::endl;
    }

    return 0;
}
```

36. Write a program that asks the user for three numbers and displays the largest one.

```
#include <iostream>

int main() {
    double num1, num2, num3;

    // Prompt the user for three numbers
    std::cout << "Enter the first number: ";
    std::cin >> num1;

    std::cout << "Enter the second number: ";
    std::cin >> num2;
```

```
    std::cout << "Enter the third number: ";
    std::cin >> num3;

    // Determine and display the largest number
    if (num1 >= num2 && num1 >= num3) {
        std::cout << "The largest number is: " << num1
<< std::endl;
    } else if (num2 >= num1 && num2 >= num3) {
        std::cout << "The largest number is: " << num2
<< std::endl;
    } else {
        std::cout << "The largest number is: " << num3
<< std::endl;
    }

    return 0;
}
```

37. Make a program that reads three numbers, and informs if their sum is divisible by 5 or not.

```
#include <iostream>

int main() {
   double num1, num2, num3;

    // Prompt the user for three numbers
   std::cout << "Enter the first number: ";
   std::cin >> num1;

    std::cout << "Enter the second number: ";
   std::cin >> num2;

    std::cout << "Enter the third number: ";
   std::cin >> num3;

    // Calculate the sum of the three numbers
   double sum = num1 + num2 + num3;

    // Check if the sum is divisible by 5
```

```
  if (static_cast<int>(sum) % 5 == 0) {
     std::cout << "The sum of the numbers is divisi-
ble by 5." << std::endl;
  } else {
     std::cout << "The sum of the numbers is not
divisible by 5." << std::endl;
  }

    return 0;
}
```

38. Write a program that asks for an integer and checks if it is divisible by 3 and 5 at the same time.

```
#include <iostream>

int main() {
  int number;

    // Prompt the user for an integer
  std::cout << "Enter an integer: ";
  std::cin >> number;
```

```
    // Check if the number is divisible by both 3 and
5
  if (number % 3 == 0 && number % 5 == 0) {
     std::cout << "The number is divisible by both 3
and 5." << std::endl;
  } else {
     std::cout << "The number is not divisible by
both 3 and 5." << std::endl;
  }

    return 0;
}
```

39. Make a program that asks for two numbers and displays if the first is divisible by the second

```
#include <iostream>

int main() {
  double num1, num2;
```

```
    // Prompt the user for two numbers
    std::cout << "Enter the first number: ";
    std::cin >> num1;

    std::cout << "Enter the second number: ";
    std::cin >> num2;

    // Ensure the second number isn't zero before attempting division
    if (num2 == 0) {
        std::cout << "The second number cannot be zero (division by zero is undefined)." << std::endl;
    } else if (static_cast<int>(num1) % static_cast<int>(num2) == 0) {
        std::cout << "The first number is divisible by the second number." << std::endl;
    } else {
        std::cout << "The first number is not divisible by the second number." << std::endl;
    }

    return 0;
```

```
}
```

40. Make a program that reads the scores of two tests and reports whether the student passed (score greater than or equal to 6) or failed (score less than 6) in each of the tests.

```
#include <iostream>

int main() {
   double test1Score, test2Score;

    // Prompt the user for scores of two tests
   std::cout << "Enter the score for the first test: ";
   std::cin >> test1Score;

    std::cout << "Enter the score for the second test:
";
   std::cin >> test2Score;

    // Determine and display the result for the first
```

```
test
  if (test1Score >= 6) {
    std::cout << "Passed in the first test." <<
std::endl;
  } else {
    std::cout << "Failed in the first test." <<
std::endl;
  }

   // Determine and display the result for the
second test
  if (test2Score >= 6) {
    std::cout << "Passed in the second test." <<
std::endl;
  } else {
    std::cout << "Failed in the second test." <<
std::endl;
  }

   return 0;
}
```

41. Make a program that reads the grades of two tests, calculates the simple arithmetic mean, and informs whether the student passed (average greater than or equal to 6) or failed (average less than 6).

```
#include <iostream>

int main() {
   double test1Grade, test2Grade;

    // Prompt the user for grades of two tests
   std::cout << "Enter the grade for the first test: ";
   std::cin >> test1Grade;

    std::cout << "Enter the grade for the second test:
";
   std::cin >> test2Grade;

    // Calculate the simple arithmetic mean of the
```

```
grades
    double average = (test1Grade + test2Grade) / 2;

     // Display the average
    std::cout << "Average grade: " << average <<
std::endl;

     // Determine and display the result based on the
average
    if (average >= 6) {
        std::cout << "Student passed." << std::endl;
    } else {
        std::cout << "Student failed." << std::endl;
    }

     return 0;
}
```

42. Make a program that reads the age of three people and how many of them are of legal age (age 18 or older).

```
#include <iostream>

int main() {
  int age1, age2, age3;
  int countLegalAge = 0;

   // Prompt the user for the ages of three people
  std::cout << "Enter the age of the first person: ";
  std::cin >> age1;
  if (age1 >= 18) countLegalAge++;

   std::cout << "Enter the age of the second person:
";
  std::cin >> age2;
  if (age2 >= 18) countLegalAge++;

   std::cout << "Enter the age of the third person: ";
  std::cin >> age3;
```

```
    if (age3 >= 18) countLegalAge++;

     // Display the result
    std::cout << countLegalAge << " of them are of
legal age (18 or older)." << std::endl;

     return 0;
}
```

43. Make a program that reads three numbers, and displays them on the screen in ascending order.

```
#include <iostream>

int main() {
    double num1, num2, num3;

     // Prompt the user for three numbers
    std::cout << "Enter the first number: ";
    std::cin >> num1;

     std::cout << "Enter the second number: ";
```

```
std::cin >> num2;

  std::cout << "Enter the third number: ";
std::cin >> num3;

  // Display the numbers in ascending order
if (num1 <= num2 && num1 <= num3) {
   std::cout << num1 << ", ";
   if (num2 <= num3) {
      std::cout << num2 << ", " << num3;
   } else {
      std::cout << num3 << ", " << num2;
   }
} else if (num2 <= num1 && num2 <= num3) {
   std::cout << num2 << ", ";
   if (num1 <= num3) {
      std::cout << num1 << ", " << num3;
   } else {
      std::cout << num3 << ", " << num1;
   }
} else {
   std::cout << num3 << ", ";
   if (num1 <= num2) {
```

```
        std::cout << num1 << "," << num2;
    } else {
        std::cout << num2 << "," << num1;
    }
  }

    std::cout << std::endl;

    return 0;
}
```

44. Write a program that reads three numbers and tells you if they can be the sides of a triangle (the sum of two sides must always be greater than the third side).

```
#include <iostream>

int main() {
  double side1, side2, side3;
```

```
    // Prompt the user for three sides
    std::cout << "Enter the first side: ";
    std::cin >> side1;

    std::cout << "Enter the second side: ";
    std::cin >> side2;

    std::cout << "Enter the third side: ";
    std::cin >> side3;

    // Check the triangle inequality conditions
    if (side1 + side2 > side3 && side1 + side3 > side2
&& side2 + side3 > side1) {
        std::cout << "The given sides can form a trian-
gle." << std::endl;
    } else {
        std::cout << "The given sides cannot form a tri-
angle." << std::endl;
    }

    return 0;
}
```

45. Make a program that reads the year of birth of a person and informs if he is able to vote (age greater than or equal to 16 years old).

```
#include <iostream>
#include <ctime>  // For getting the current year

int main() {
   int birthYear;
   int currentYear;

    // Get the current year
   time_t now = time(0);
   tm *ltm = localtime(&now);
   currentYear = 1900 + ltm->tm_year;

    // Prompt the user for their year of birth
   std::cout << "Enter your year of birth: ";
   std::cin >> birthYear;

    int age = currentYear - birthYear;
```

```
    // Check if the person is eligible to vote
   if (age >= 16) {
      std::cout << "You are eligible to vote." <<
std::endl;
   } else {
      std::cout << "You are not eligible to vote." <<
std::endl;
   }

    return 0;
}
```

46. Create a program that asks for a person's age and displays whether they are a child (0-12 years old), teenager (13-17 years old), adult (18-59 years old), or elderly (60 years old or older), using nested conditionals, without using logical operators.

```
#include <iostream>

int main() {
  int age;

    // Prompt the user for their age
  std::cout << "Enter your age: ";
  std::cin >> age;

    // Check the age range using nested condition-
als
  if (age < 13) {
     std::cout << "You are a child." << std::endl;
```

```
    } else if (age < 18) {
        std::cout << "You are a teenager." << std::endl;
    } else if (age < 60) {
        std::cout << "You are an adult." << std::endl;
    } else {
        std::cout << "You are elderly." << std::endl;
    }

    return 0;
}
```

47. Create a program that asks for a person's age and displays whether they are a child (0-12 years old), teenager (13-17 years old), adult (18-59 years old), or elderly (60 years old or older), using logical operators, without using else, elif, etc.

```
#include <iostream>
```

```cpp
int main() {
  int age;

   // Prompt the user for their age
  std::cout << "Enter your age: ";
  std::cin >> age;

   // Check the age range using only if statements
and logical operators
  if (age >= 0 && age <= 12) {
    std::cout << "You are a child." << std::endl;
  }
  if (age >= 13 && age <= 17) {
    std::cout << "You are a teenager." << std::endl;
  }
  if (age >= 18 && age <= 59) {
    std::cout << "You are an adult." << std::endl;
  }
  if (age >= 60) {
    std::cout << "You are elderly." << std::endl;
  }
```

```
    return 0;
}
```

48. Make a program that reads a person's age and informs if he is not able to vote (age less than 16 years old), if he is able to vote but is not obligated (16, 17 years old, or age equal to or greater than 70 years), or if it is obligatory (18 to 69 years old).

```
#include <iostream>

int main() {
  int age;

    // Prompt the user for their age
  std::cout << "Enter your age: ";
  std::cin >> age;

    // Check the age and display the corresponding
voting status
```

```
  if (age < 16) {
      std::cout << "You are not able to vote." <<
std::endl;
  } else if (age == 16 || age == 17 || age >= 70) {
      std::cout << "You are able to vote, but it's not
obligatory." << std::endl;
  } else {
      std::cout << "Voting is obligatory for you." <<
std::endl;
  }

    return 0;
}
```

49. Make a program that reads three grades from a student and reports whether he passed (final grade greater than or equal to 7), failed (final grade less than 4) or was in recovery (final grade between 4 and 7).

```
#include <iostream>

int main() {
   double grade1, grade2, grade3, average;

    // Read the three grades from the user
   std::cout << "Enter the first grade: ";
   std::cin >> grade1;


    std::cout << "Enter the second grade: ";
   std::cin >> grade2;

    std::cout << "Enter the third grade: ";
```

```
    std::cin >> grade3;

    // Calculate the average of the grades
    average = (grade1 + grade2 + grade3) / 3.0;

    // Determine and display the student's status
    if (average >= 7) {
        std::cout << "You passed!" << std::endl;
    } else if (average < 4) {
        std::cout << "You failed." << std::endl;
    } else {
        std::cout << "You are in recovery." << std::endl;
    }

    return 0;
}
```

50. Write a program that asks for a person's height and weight and calculates their body mass index (BMI), displaying the corresponding category (underweight, normal weight, overweight, obese, severely obese).

```
#include <iostream>

int main() {
   double height, weight, bmi;

    // Get height and weight from the user
   std::cout << "Enter your height in meters: ";
   std::cin >> height;


    std::cout << "Enter your weight in kilograms: ";
   std::cin >> weight;

    // Calculate BMI
```

```
    bmi = weight / (height * height);

    // Display BMI and category
    std::cout << "Your BMI is: " << bmi << std::endl;

    if (bmi < 18.5) {
        std::cout << "You are underweight." <<
std::endl;
    } else if (bmi >= 18.5 && bmi < 24.9) {
        std::cout << "You are of normal weight." <<
std::endl;
    } else if (bmi >= 24.9 && bmi < 30) {
        std::cout << "You are overweight." << std::endl;
    } else if (bmi >= 30 && bmi < 35) {
        std::cout << "You are obese." << std::endl;
    } else {
        std::cout << "You are severely obese." <<
std::endl;
    }

    return 0;
}
```

51. Write a program that asks the user to enter their grade (0-100) and prints their letter grade (A for 90-100, B for 80-89, C for 70-79, D for 60-69, F for under 60).

```
#include <iostream>

int main() {
  int grade;

   // Ask user for their grade
  std::cout << "Enter your grade (0-100): ";
  std::cin >> grade;

   // Determine and print letter grade
  if (grade >= 90 && grade <= 100) {
    std::cout << "Your letter grade is: A" << std::endl;
  } else if (grade >= 80 && grade < 90) {
    std::cout << "Your letter grade is: B" << std::endl;
  } else if (grade >= 70 && grade < 80) {
    std::cout << "Your letter grade is: C" << std::endl;
  } else if (grade >= 60 && grade < 70) {
```

```
    std::cout << "Your letter grade is: D" << std::endl;
  } else if (grade < 60) {
    std::cout << "Your letter grade is: F" << std::endl;
  } else {
    std::cout << "Invalid grade input." << std::endl;
  }

  return 0;
}
```

52. Write a program that reads the values *a*, *b* and *c* of a quadratic equation, and says if the roots of the function are real or imaginary.

```
#include <iostream>

int main() {
  double a, b, c, discriminant;

  // Read coefficients
  std::cout << "Enter the value of a: ";
```

```
  std::cin >> a;

   std::cout << "Enter the value of b: ";
  std::cin >> b;

   std::cout << "Enter the value of c: ";
  std::cin >> c;

   // Calculate the discriminant
  discriminant = b * b - 4 * a * c;

   // Check the nature of roots based on the dis-
criminant
  if (discriminant > 0) {
     std::cout << "The roots are real and distinct." <<
std::endl;
  } else if (discriminant == 0) {
     std::cout << "The roots are real and the same."
<< std::endl;
  } else {
     std::cout << "The roots are imaginary." <<
std::endl;
  }
```

```
    return 0;
}
```

53. Create a program that reads the price of a product. If the price is more than US $200, the tax is 5%, if it is more than US $500, the tax is 7.5%. Products up to US $200 do not pay taxes. Print the final price of the product on the screen.

```
#include <iostream>
#include <iomanip>

int main() {
   double price, tax, finalPrice;

    // Read product price
   std::cout << "Enter the price of the product: $";
```

```
    std::cin >> price;

    // Determine tax rate based on price
    if (price > 500) {
        tax = 0.075; // 7.5%
    } else if (price > 200) {
        tax = 0.05;  // 5%
    } else {
        tax = 0;    // No tax
    }

    // Calculate final price
    finalPrice = price + (price * tax);

    // Print the final price
    std::cout << std::fixed << std::setpreci-
sion(2);
    std::cout << "The final price of the product
(including tax) is: $" << finalPrice << std::endl;

    return 0;
```

```
}
```

54. Write a program that reads a year (a four digit number) and checks whether it is a leap year or not.

```
#include <iostream>

int main() {
  int year;

    // Prompt the user for input
  std::cout << "Enter a year (four digits): ";
  std::cin >> year;

    // Check if the year is a leap year
  if ((year % 4 == 0 && year % 100 != 0) || year %
400 == 0) {
     std::cout << year << " is a leap year." << std::endl;
  } else {
     std::cout << year << " is not a leap year." <<
std::endl;
```

```
    }

    return 0;
}
```

55. Input three numbers representing the coefficients of a quadratic equation (a, b, c) and find its roots, also mention if they are real or complex

```
#include <iostream>
#include <cmath>

int main() {
    double a, b, c, discriminant, realPart, imaginaryPart, root1, root2;

    // Read the coefficients of the quadratic equation
    std::cout << "Enter the coefficients a, b and c separated by spaces: ";
    std::cin >> a >> b >> c;
```

```
    // Calculate the discriminant
    discriminant = b * b - 4 * a * c;

    if (discriminant > 0) {
        // Two real and distinct roots
        root1 = (-b + sqrt(discriminant)) / (2 * a);
        root2 = (-b - sqrt(discriminant)) / (2 * a);
        std::cout << "Roots are real and distinct." <<
std::endl;
        std::cout << "Root 1: " << root1 << std::endl;
        std::cout << "Root 2: " << root2 << std::endl;
    } else if (discriminant == 0) {
        // Two real and equal roots
        root1 = -b / (2 * a);
        std::cout << "Roots are real and equal." <<
std::endl;
        std::cout << "Root 1 and Root 2: " << root1 <<
std::endl;
    } else {
        // Complex roots
        realPart = -b / (2 * a);
        imaginaryPart = sqrt(-discriminant) / (2 * a);
```

```
        std::cout << "Roots are complex and distinct."
<< std::endl;
        std::cout << "Root 1: " << realPart << " + " <<
imaginaryPart << "i" << std::endl;
        std::cout << "Root 2: " << realPart << " - " <<
imaginaryPart << "i" << std::endl;
    }

    return 0;
}
```

Repeat Loops

56. Write a program that displays the numbers 1 through 10 using a while loop.

```
#include <iostream>

int main() {
  int i = 1;  // Initialize counter to 1

    while (i <= 10) {
     std::cout << i << std::endl;
     i++;  // Increment the counter
  }

    return 0;
}
```

57. Write a program that displays the numbers 1 through 10 using a for loop.

```
#include <iostream>

int main() {
  for (int i = 1; i <= 10; i++) {
    std::cout << i << std::endl;
  }

   return 0;
}
```

58. Write a program that displays all numbers from 1 to 100, using a while loop.

```
#include <iostream>

int main() {
  int i = 1;  // Initialize counter to 1

   while (i <= 100) {
```

```
        std::cout << i << " ";
        i++;  // Increment the counter
    }

    return 0;
}
```

59. Write a program that displays all numbers from 1 to 100, using a for loop.

```
#include <iostream>

int main() {
    for (int i = 1; i <= 100; i++) {
        std::cout << i << " ";
    }

    return 0;
}
```

60. Write a program that displays all numbers from 1 to 100, using a do-while loop.

```
#include <iostream>

int main() {
  int i = 1;

    do {
     std::cout << i << " ";
     i++;
  } while (i <= 100);

    return 0;
}
```

61. Write a program that prints all even numbers from 1 to 100 using a while loop and an if statement inside the loop to check if the number is even or not.

```
#include <iostream>

int main() {
  int i = 1;

    while (i <= 100) {
     if (i % 2 == 0) {  // Check if the number is even
        std::cout << i << " ";
     }
     i++;
  }

    return 0;
}
```

62. Write a program that prints all even numbers from 1 to 100 using a while loop without using any if statement.

```
#include <iostream>

int main() {
  int i = 2;

    while (i <= 100) {
     std::cout << i << " ";
     i += 2; // Increment by 2 to get the next even
number
  }

    return 0;
}
```

63. Write a program that displays even numbers 1 to 50 and odd numbers 51 to 100 using a repeating loop.

```
#include <iostream>

int main() {
   // Display even numbers from 1 to 50
   for (int i = 2; i <= 50; i += 2) {
      std::cout << i << " ";
   }

    std::cout << std::endl; // For better separation

    // Display odd numbers from 51 to 100
   for (int i = 51; i <= 100; i += 2) {
      std::cout << i << " ";
   }

    return 0;
}
```

64. Create a program that prompts the user for a number and displays the table of that number using a loop.

```
#include <iostream>

int main() {
  int number;

    // Ask user for a number
  std::cout << "Enter a number: ";
  std::cin >> number;

    // Display the table of that number
  for (int i = 1; i <= 10; i++) {
     std::cout << number << " x " << i << " = " << num-
ber * i << std::endl;
  }

    return 0;
}
```

65. Create a program that displays the table of all numbers from 1 to 10.

```
#include <iostream>

int main() {


    for (int number = 1; number <= 10; number++) {
     std::cout << "Table of " << number << ":" <<
std::endl;


        for (int i = 1; i <= 10; i++) {
       std::cout << number << " x " << i << " = " <<
number * i << std::endl;
     }

        std::cout << std::endl;  // To separate tables
  }

    return 0;
}
```

66. Write a program that calculates and displays the value of the power of a number entered by the user raised to an exponent also entered by the user, using repetition loops.

```
#include <iostream>

int main() {
   double base, result = 1.0;
   int exponent;

    // Get input from the user
   std::cout << "Enter the base: ";
   std::cin >> base;

    std::cout << "Enter the exponent (positive or
negative integer): ";
   std::cin >> exponent;

    if (exponent > 0) {
     for (int i = 1; i <= exponent; i++) {
```

```
        result *= base;
      }
    } else if (exponent < 0) {
      for (int i = 1; i <= -exponent; i++) {
        result /= base;
      }
    } // If exponent is 0, result remains 1.0

    std::cout << base << " raised to the power " <<
exponent << " is: " << result << std::endl;

    return 0;
}
```

67. Create a program that displays the first N first perfect squares, where N is informed by the user, using a loop.

```
#include <iostream>

int main() {
  int N;
```

```
    // Get input from the user
   std::cout << "Enter the value of N: ";
   std::cin >> N;

    for (int i = 1; i <= N; i++) {
     std::cout << "Square of " << i << " is: " << i * i <<
std::endl;
   }

    return 0;
}
```

68. Write a program that asks the user for a number N and says whether it is prime or not.

```
#include <iostream>

int main() {
   int N, i;

    // Get input from the user
```

```
    std::cout << "Enter a number N: ";
    std::cin >> N;

     if (N <= 1) {
      std::cout << N << " is not a prime number." <<
std::endl;
      return 0;
    }

     for (i = 2; i * i <= N; i++) {
      if (N % i == 0) {
         std::cout << N << " is not a prime number." <<
std::endl;
         return 0;  // If any divisor found, break the
loop and print non-prime message
      }
    }


     std::cout << N << " is a prime number." <<
std::endl;
    return 0;
}
```

69. Write a program that prompts the user for a number N and displays all prime numbers less than N.

```
#include <iostream>

int main() {
  int N;

   // Get input from the user
  std::cout << "Enter a number N: ";
  std::cin >> N;

   // Print prime numbers less than N
  std::cout << "Prime numbers less than " << N << "
are: ";

   for (int num = 2; num < N; num++) {
    bool isPrime = true;

      for (int i = 2; i * i <= num; i++) {
     if (num % i == 0) {
       isPrime = false;
```

```
            break;  // If any divisor found, set isPrime to
false and break the inner loop
        }
    }

        if (isPrime) {
        std::cout << num << " ";
    }
  }

    std::cout << std::endl;  // Newline after printing
all prime numbers
  return 0;
}
```

70. Create a program that displays the first N prime numbers, where N is informed by the user, using a loop.

```
#include <iostream>

int main() {
  int N;
```

```
    // Get input from the user
   std::cout << "Enter the number of prime numbers
you want: ";
   std::cin >> N;

    std::cout << "The first " << N << " prime numbers
are: ";

    int count = 0;  // to keep track of the number of
primes found
   for (int num = 2; count < N; num++) {
     bool isPrime = true;

        for (int i = 2; i * i <= num; i++) {
       if (num % i == 0) {
         isPrime = false;
         break;  // If any divisor found, set isPrime to
false and break the inner loop
       }
     }

        if (isPrime) {
       std::cout << num << " ";
```

```
            count++;
        }
    }

    std::cout << std::endl;  // Newline after printing all prime numbers
    return 0;
}
```

71. Write a program that prompts the user for two numbers A and B and displays all numbers between A and B.

```
#include <iostream>

int main() {
    int A, B;

    // Get the two numbers from the user
    std::cout << "Enter the value of A: ";
    std::cin >> A;

    std::cout << "Enter the value of B: ";
```

```
    std::cin >> B;

    if (A > B) {
        std::cout << "A should be less than or equal to B.
\n";
        return 1;  // Exit with error code
    }

    // Display numbers between A and B
    std::cout << "Numbers between " << A << " and "
<< B << " are:\n";

    for (int i = A; i <= B; i++) {
        std::cout << i << " ";
    }

    std::cout << std::endl;  // Newline after printing
all numbers
    return 0;
}
```

72. Write a program that reads numbers from the user until a negative number is entered, and prints the sum of the positive numbers.

```
#include <iostream>

int main() {
   double number, total = 0;

    std::cout << "Enter numbers (enter a negative number to stop): " << std::endl;

    while (true) {
     std::cin >> number;

        if (number < 0) {
       break;  // Exit the loop if a negative number is entered
     }

        total += number;
```

```
    }

    std::cout << "Sum of the positive numbers: " <<
total << std::endl;

    return 0;
}
```

73. Write a program that asks the user for a number N and displays the sum of all numbers from 1 to N.

```
#include <iostream>

int main() {
    int N, sum = 0;

    std::cout << "Enter a number N: ";
    std::cin >> N;

    for(int i = 1; i <= N; i++) {
     sum += i;
    }
```

```
    std::cout << "The sum of numbers from 1 to " <<
N << " is: " << sum << std::endl;

    return 0;
}
```

74. Write a program that calculates and displays the sum of even numbers from 1 to 100 using a repeating loop.

```
#include <iostream>

int main() {
  int sum = 0;

    for (int i = 2; i <= 100; i += 2) {
     sum += i;
  }

    std::cout << "The sum of even numbers from 1
to 100 is: " << sum << std::endl;

    return 0;
```

```
}
```

75. Write a program that prompts the user for a number and displays the Fibonacci sequence up to the given number using a repeating loop.

```
#include <iostream>

int main() {
  int n, t1 = 0, t2 = 1, nextTerm = 0;

    std::cout << "Enter the number of terms: ";
  std::cin >> n;

    std::cout << "Fibonacci Series: ";

    for (int i = 1; i <= n; ++i) {
     if(i == 1) {
        std::cout << t1 << " ";
        continue;
     }
     if(i == 2) {
```

```
            std::cout << t2 << " ";
            continue;
        }
        nextTerm = t1 + t2;
        t1 = t2;
        t2 = nextTerm;


            std::cout << nextTerm << " ";
    }
    std::cout << std::endl;

     return 0;
}
```

76. Write a program that reads numbers from the user until zero is entered, and displays the average of the numbers entered.

```
#include <iostream>
```

```
int main() {
   double sum = 0.0;
   int count = 0;
   double num;

    std::cout << "Enter numbers (enter 0 to stop): "
<< std::endl;

    do {
     std::cin >> num;
     if (num != 0) {
        sum += num;
        count++;
     }
   } while (num != 0);

    if (count != 0) {
     double average = sum / count;
     std::cout << "Average of the numbers entered: "
<< average << std::endl;
   } else {
     std::cout << "No numbers were entered." <<
```

```
std::endl;
   }

    return 0;
}
```

77. Write a program that prompts the user for a list of numbers, until the user types the number zero, and displays the largest and smallest numbers in the list.

```
#include <iostream>
#include <limits>  // For std::numeric_limits

int main() {
   double largest = std::numeric_limits<dou-
ble>::lowest();
   double smallest = std::numeric_limits<double>::
max();
   double num;

    std::cout << "Enter numbers (enter 0 to stop): "
```

```
<< std::endl;

    do {
     std::cin >> num;

        if (num != 0) {
        if (num > largest) {
          largest = num;
        }
        if (num < smallest) {
          smallest = num;
        }
      }
  } while (num != 0);

    if (largest != std::numeric_limits<double>::low-
est() && smallest != std::numeric_limits<dou-
ble>::max()) {
      std::cout << "Largest number: " << largest <<
std::endl;
      std::cout << "Smallest number: " << smallest <<
std::endl;
  } else {
```

```
    std::cout << "No numbers were entered." <<
std::endl;
  }

   return 0;
}
```

78. Write a program that prompts the user for a number and displays its divisors.

```
#include <iostream>

int main() {
  int num;

   std::cout << "Enter a number: ";
  std::cin >> num;

   std::cout << "Divisors of " << num << " are: ";

   for (int i = 1; i <= num; i++) {
    if (num % i == 0) {
      std::cout << i << " ";
    }
```

```
    }

    std::cout << std::endl;

    return 0;
}
```

79. Write a program that determines the greatest common divisor (GCD) between two numbers entered by the user.

```
#include <iostream>

int main() {
    int a, b;

    // Get input from user
    std::cout << "Enter two integers: ";
    std::cin >> a >> b;

    // Calculate GCD using the Euclidean algorithm
    while (b != 0) {
        int remainder = a % b;
```

```
    a = b;
    b = remainder;
  }

  // Print the result
  std::cout << "The GCD of the numbers is: " << a <<
std::endl;

  return 0;
}
```

80. Write a program that determines the lowest common multiple (LCM) between two numbers entered by the user.

```
#include <iostream>

int main() {
  int a, b, lcm, gcd, temp_a, temp_b;

  // Get input from user
  std::cout << "Enter two integers: ";
  std::cin >> a >> b;
```

```
    // Store the original values for later use
    temp_a = a;
    temp_b = b;

    // Calculate GCD using the Euclidean algorithm
    while (b != 0) {
        int remainder = a % b;
        a = b;
        b = remainder;
    }

    gcd = a;

    // Calculate LCM
    lcm = (temp_a * temp_b) / gcd;

    // Print the result
    std::cout << "The LCM of " << temp_a << " and " <<
temp_b << " is " << lcm << "." << std::endl;

    return 0;
}
```

81. Write a program that calculates the series below up to the tenth element:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

```
#include <iostream>

int main() {
   double x;
   double e_to_x = 1.0;  // Initialize with the first term (1)

    // Get input from user
   std::cout << "Enter the value of x: ";
   std::cin >> x;

    double power_of_x = x;  // Initialize with x for the first term
   double factorial = 1.0;  // Initialize with 1 for the first term

    // Calculate e^x using the series up to the tenth element
```

```
  for (int i = 1; i <= 10; i++) {
    e_to_x += power_of_x / factorial;
    power_of_x *= x;      // Calculate x^(i+1)
    factorial *= (i + 1);  // Calculate (i+1)!
  }

   // Print the result
  std::cout << "The value of e^" << x << " using the
series up to the tenth element is: " << e_to_x <<
std::endl;

    return 0;
}
```

82. Rewrite the previous exercise code until the difference between the terms is less than 0.001.

```
#include <iostream>

int main() {
  double x;
  double e_to_x = 1.0;  // Initialize with the first
```

```
term (1)

    // Get input from user
   std::cout << "Enter the value of x: ";
   std::cin >> x;

    double power_of_x = x;  // Initialize with x for
the first term
   double factorial = 1.0;  // Initialize with 1 for the
first term
   double term = power_of_x / factorial;  // value of
the term

    // Calculate e^x using the series until the differ-
ence between terms is less than 0.001
   while (term >= 0.001) {
      e_to_x += term;
      factorial *= (factorial + 1);  // Calculate (i+1)!
      power_of_x *= x;      // Calculate x^(i+1)
      term = power_of_x / factorial;  // Calculate the
new term
   }
```

```
    // Print the result
   std::cout << "The approximated value of e^" << x
<< " using the series until the difference between
terms is less than 0.001 is: " << e_to_x << std::endl;

    return 0;
}
```

83. Make a program that calculates the value of sine using the Taylor series according to the equation below until the difference between the terms is less than 0.001.

$$sen(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \cdots$$

```
#include <iostream>
#include <cmath>

int main() {
   double x;
```

```
  double sine_of_x = 0.0;

   // Get input from user
  std::cout << "Enter the value of x (in radians): ";
  std::cin >> x;

   double power_of_x = x;  // Initialize with x for the first term
  double factorial = 1.0;  // Initialize with 1 for the first term
  double term = power_of_x;  // value of the first term
  int n = 0;  // start with n = 0

   // Calculate sin(x) using the Taylor series until the difference between terms is less than 0.001
  while (fabs(term) >= 0.001) {
    sine_of_x += term;  // add the term to the result
    n++;  // increase n by 1
    factorial *= (2*n) * (2*n + 1);  // Calculate (2n+1)! = (2n)! * (2n+1)
    power_of_x *= x * x;  // Calculate x^(2n+1)
    term = power_of_x / factorial;  // Calculate the
```

```
new term
    if (n % 2 != 0) {  // alternate the sign of the term
      term = -term;
    }
  }

   // Print the result
  std::cout << "The approximated value of sin(" <<
x << ") using the Taylor series until the difference
between terms is less than 0.001 is: " << sine_of_x
<< std::endl;
```

84. Make a program that calculates the value of cosine using the Taylor series according to the equation below until the difference between the terms is less than 0.001.

$$cos(x) = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \frac{x^{10}}{10!} \cdots$$

```
#include <iostream>
#include <cmath>
```

```
int main() {
   double x;
   double cosine_of_x = 1.0;

    // Get input from user
   std::cout << "Enter the value of x (in radians): ";
   std::cin >> x;

    double power_of_x = 1.0; // Initialize with 1 for
the first term
   double factorial = 1.0; // Initialize with 1 for the
first term
   double term = 1.0; // value of the first term
   int n = 0; // start with n = 0

    // Calculate cos(x) using the Taylor series until
the difference between terms is less than 0.001
   while (fabs(term) >= 0.001) {
      n++; // increase n by 1
      power_of_x *= x * x; // Calculate x^(2n)
      factorial *= (2*n - 1) * (2*n); // Calculate (2n)!
      term = power_of_x / factorial; // Calculate the
```

```
new term
    if (n % 2 != 0) {  // alternate the sign of the term
      term = -term;
    }
    cosine_of_x += term;  // add the term to the
result
  }

   // Print the result
  std::cout << "The approximated value of cos("
<< x << ") using the Taylor series until the differ-
ence between terms is less than 0.001 is: " <<
cosine_of_x << std::endl;

    return 0;
}
```

85. Write a program that displays the sine and cosine value of all numbers from 0 to 6.3, with a step of 0.1, using Taylor series to calculate the respective sines and cosines.

```
#include <iostream>
#include <cmath>

int main() {
   const double STEP_SIZE = 0.1;
   const double END_POINT = 6.3;


    for(double x = 0; x <= END_POINT; x += STEP_
SIZE) {
      // Calculate sine using Taylor series
      double sine_of_x = x;
      double term = x;
      double power_of_x = x;
      double factorial = 1;
```

```
int n = 1;
while (fabs(term) >= 0.001) {
  power_of_x *= x * x;
  factorial *= (2 * n) * (2 * n + 1);
  term = power_of_x / factorial;
  if (n % 2 != 0) {
    term = -term;
  }
  sine_of_x += term;
  n++;
}


  // Calculate cosine using Taylor series
double cosine_of_x = 1.0;
term = 1.0;
power_of_x = 1.0;
factorial = 1.0;
n = 0;
while (fabs(term) >= 0.001) {
  n++;
  power_of_x *= x * x;
```

```
        factorial *= (2 * n - 1) * (2 * n);
        term = power_of_x / factorial;
        if (n % 2 != 0) {
           term = -term;
        }
        cosine_of_x += term;
      }

         // Print sine and cosine values
      std::cout << "For x = " << x << ", sine(x) = " <<
sine_of_x << " and cosine(x) = " << cosine_of_x <<
std::endl;
   }


    return 0;
}
```

Arrays

86. Write a program that reads an array of integers and displays the elements in reverse order.

```
#include <iostream>

int main() {
  int n;

   // Prompt the user for the size of the array
  std::cout << "Enter the number of elements: ";
  std::cin >> n;

   // Allocate memory for the array
  int* arr = new int[n];

   // Prompt the user for the array elements
  std::cout << "Enter " << n << " integers:" <<
std::endl;
```

```
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    // Display the elements in reverse order
    std::cout << "Elements in reverse order:" <<
std::endl;
    for (int i = n - 1; i >= 0; i--) {
        std::cout << arr[i] << " ";
    }
    std::cout << std::endl;

    // Free the allocated memory
    delete[] arr;

    return 0;
}
```

87. Create a program that reads an array of integers and displays the sum of all the elements.

```
#include <iostream>

int main() {
   int n, sum = 0;

    // Prompt the user for the size of the array
   std::cout << "Enter the number of elements: ";
   std::cin >> n;

    // Allocate memory for the array
   int* arr = new int[n];

    // Prompt the user for the array elements
   std::cout << "Enter " << n << " integers:" <<
std::endl;
   for (int i = 0; i < n; i++) {
      std::cin >> arr[i];
   }
```

```
    // Calculate the sum of the elements
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

    // Display the sum
    std::cout << "The sum of the elements is: " << sum
<< std::endl;

    // Free the allocated memory
    delete[] arr;

    return 0;
}
```

88. Write a program that reads an array of integers and displays the average of the elements.

```
#include <iostream>

int main() {
```

```
    int n;
    double sum = 0.0;  // Use double to handle possible fractional results

    // Prompt the user for the size of the array
    std::cout << "Enter the number of elements: ";
    std::cin >> n;

    // Allocate memory for the array
    int* arr = new int[n];

    // Prompt the user for the array elements
    std::cout << "Enter " << n << " integers:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    // Calculate the sum of the elements
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }

    // Calculate the average
```

```
    double average = sum / n;

    // Display the average
    std::cout << "The average of the elements is: " <<
average << std::endl;

    // Free the allocated memory
    delete[] arr;

    return 0;
}
```

89. Write a program that reads an array of integers and displays the largest element in the array.

```
#include <iostream>

int main() {
    int n;

    // Prompt the user for the size of the array
    std::cout << "Enter the number of elements: ";
```

```
    std::cin >> n;

    // Allocate memory for the array
    int* arr = new int[n];

    // Prompt the user for the array elements
    std::cout << "Enter " << n << " integers:" <<
std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    // Find the largest element
    int max = arr[0];
    for (int i = 1; i < n; i++) {
        if (arr[i] > max) {
            max = arr[i];
        }
    }

    // Display the largest element
    std::cout << "The largest element is: " << max <<
std::endl;
```

```
    // Free the allocated memory
    delete[] arr;

    return 0;
}
```

90. Create a program that reads two vectors of integers of the same size and displays a new vector with the sum of the corresponding elements of the two vectors.

```
#include <iostream>

int main() {
    int n;

    // Prompt the user for the size of the vectors
    std::cout << "Enter the number of elements: ";
    std::cin >> n;

    // Allocate memory for the vectors
```

```
    int* vec1 = new int[n];
    int* vec2 = new int[n];
    int* sumVec = new int[n];

     // Prompt the user for the elements of the first vector
    std::cout << "Enter elements of the first vector:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> vec1[i];
    }

     // Prompt the user for the elements of the second vector
    std::cout << "Enter elements of the second vector:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> vec2[i];
    }

     // Calculate the sum of corresponding elements
    for (int i = 0; i < n; i++) {
        sumVec[i] = vec1[i] + vec2[i];
```

```
    }

    // Display the resulting vector
    std::cout << "Resulting vector:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cout << sumVec[i] << " ";
    }
    std::cout << std::endl;

    // Free the allocated memory
    delete[] vec1;
    delete[] vec2;
    delete[] sumVec;

    return 0;
}
```

91. Write a program that reads two arrays of integers with the same size and displays a new array with the elements resulting from the multiplication of the corresponding elements of the two arrays.

```
#include <iostream>

int main() {
  int n;

   // Prompt the user for the size of the arrays
  std::cout << "Enter the number of elements: ";
  std::cin >> n;

   // Allocate memory for the arrays
  int* arr1 = new int[n];
  int* arr2 = new int[n];
  int* productArr = new int[n];

   // Prompt the user for the elements of the first
array
```

```
    std::cout << "Enter elements of the first array:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> arr1[i];
    }

    // Prompt the user for the elements of the second array
    std::cout << "Enter elements of the second array:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> arr2[i];
    }

    // Calculate the product of corresponding elements
    for (int i = 0; i < n; i++) {
        productArr[i] = arr1[i] * arr2[i];
    }

    // Display the resulting array
    std::cout << "Resulting array from multiplication:" << std::endl;
```

```
    for (int i = 0; i < n; i++) {
        std::cout << productArr[i] << " ";
    }
    std::cout << std::endl;

    // Free the allocated memory
    delete[] arr1;
    delete[] arr2;
    delete[] productArr;

    return 0;
}
```

92. Write a program that reads an array of integers and displays how many times a specific number appears in the array.

```
#include <iostream>

int main() {
    int n, count = 0, searchValue;

    // Prompt the user for the size of the array
```

```
    std::cout << "Enter the number of elements: ";
    std::cin >> n;

    // Allocate memory for the array
    int* arr = new int[n];

    // Prompt the user for the elements of the array
    std::cout << "Enter the elements:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    // Prompt the user for the number to be
searched for
    std::cout << "Enter the number to search for: ";
    std::cin >> searchValue;

    // Check each element in the array
    for (int i = 0; i < n; i++) {
        if (arr[i] == searchValue) {
            count++;
        }
    }
```

```
    // Display the result
    std::cout << "The number " << searchValue <<
" appears " << count << " times in the array." <<
std::endl;

    // Free the allocated memory
    delete[] arr;

    return 0;
}
```

93. Write a program that reads an array of integers and checks if they are in ascending order.

```
#include <iostream>

int main() {
    int n;
    bool isAscending = true;

    // Prompt the user for the size of the array
```

```
std::cout << "Enter the number of elements: ";
std::cin >> n;

  // Allocate memory for the array
int* arr = new int[n];

  // Prompt the user for the elements of the array
std::cout << "Enter the elements:" << std::endl;
for (int i = 0; i < n; i++) {
   std::cin >> arr[i];
}

  // Check if the array is in ascending order
for (int i = 1; i < n; i++) {
   if (arr[i] < arr[i - 1]) {
      isAscending = false;
      break;
   }
}

  // Display the result
if (isAscending) {
   std::cout << "The array is in ascending order."
```

```
<< std::endl;
  } else {
     std::cout << "The array is not in ascending
order." << std::endl;
  }

    // Free the allocated memory
  delete[] arr;

    return 0;
}
```

94. Create a program that reads an array of integers and finds the second largest element in the array.

```
#include <iostream>
#include <limits> // for std::numeric_limits

int main() {
  int n;

    // Prompt the user for the size of the array
```

```
    std::cout << "Enter the number of elements: ";
    std::cin >> n;

    // Validate input
    if (n < 2) {
        std::cout << "You need at least two elements to
determine the second largest." << std::endl;
        return 0;
    }

    // Allocate memory for the array
    int* arr = new int[n];

    // Prompt the user for the elements of the array
    std::cout << "Enter the elements:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    int largest = std::numeric_limits<int>::min();
    int secondLargest = std::numeric_limits<in-
t>::min();

    // Find the largest and second largest elements
```

```
    for (int i = 0; i < n; i++) {
        if (arr[i] > largest) {
            secondLargest = largest;
            largest = arr[i];
        } else if (arr[i] > secondLargest && arr[i] !=
largest) {
            secondLargest = arr[i];
        }
    }

    // Display the result
    std::cout << "The second largest element is: " <<
secondLargest << std::endl;

    // Free the allocated memory
    delete[] arr;

    return 0;
}
```

95. Given an array, find the kth largest and smallest element in it.

```
#include <iostream>
#include <algorithm> // for std::sort

int main() {
  int n, k;

    // Prompt the user for the size of the array
  std::cout << "Enter the number of elements: ";
  std::cin >> n;

    // Prompt for the value of k
  std::cout << "Enter the value of k: ";
  std::cin >> k;

    // Validate input
  if (k > n || k <= 0) {
    std::cout << "Invalid value of k." << std::endl;
    return 0;
  }
```

```
    // Allocate memory for the array
    int* arr = new int[n];

    // Prompt the user for the elements of the array
    std::cout << "Enter the elements:" << std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    // Sort the array
    std::sort(arr, arr + n);

    // Display the kth smallest and kth largest elements
    std::cout << "The " << k << "th smallest element is: " << arr[k - 1] << std::endl;
    std::cout << "The " << k << "th largest element is: " << arr[n - k] << std::endl;

    // Free the allocated memory
    delete[] arr;

    return 0;
```

```
}
```

96. Write a program that checks if an array contains duplicate elements.

```
#include <iostream>

int main() {
  int n;

   // Get the size of the array
  std::cout << "Enter the number of elements: ";
  std::cin >> n;

   int arr[n];

   // Input elements
  std::cout << "Enter the elements of the array:\n";
  for (int i = 0; i < n; i++) {
    std::cin >> arr[i];
  }

   bool hasDuplicate = false;
```

```
    // Check for duplicates
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            if (arr[i] == arr[j]) {
                hasDuplicate = true;
                break;
            }
        }
        if (hasDuplicate) {
            break;
        }
    }

    if (hasDuplicate) {
        std::cout << "The array contains duplicates.\n";
    } else {
        std::cout << "The array does not contain duplicates.\n";
    }

    return 0;
}
```

97. Given an array of size n, find an element that appears more than n/2 times. (This element is called the majority element.)

```
#include <iostream>

int main() {
  int n;

   // Get the size of the array
  std::cout << "Enter the number of elements: ";
  std::cin >> n;

   int arr[n];

   // Input elements
  std::cout << "Enter the elements of the array:\n";
  for (int i = 0; i < n; i++) {
    std::cin >> arr[i];
  }
```

```
  // Boyer-Moore Voting Algorithm
int candidate = -1;
int count = 0;

  for (int i = 0; i < n; i++) {
   if (count == 0) {
      candidate = arr[i];
   }
   if (arr[i] == candidate) {
      count++;
   } else {
      count--;
   }
}

  // Verify the candidate is indeed the majority
count = 0;
for (int i = 0; i < n; i++) {
   if (arr[i] == candidate) {
      count++;
   }
}
```

```
    if (count > n / 2) {
     std::cout << "The majority element is: " << can-
didate << std::endl;
   } else {
     std::cout << "There is no majority element.\n";
   }

    return 0;
}
```

98. Create a program that reads an array of integers and checks if all elements are even.

```
#include <iostream>

int main() {
   int n;

    // Get the size of the array
   std::cout << "Enter the number of elements: ";
   std::cin >> n;
```

```
    int arr[n];

    // Input elements
    std::cout << "Enter the elements of the array:\n";
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    // Check if all elements are even
    bool allEven = true;
    for (int i = 0; i < n; i++) {
        if (arr[i] % 2 != 0) {
            allEven = false;
            break; // exit loop as soon as a non-even number is found
        }
    }

    if (allEven) {
        std::cout << "All elements in the array are even.\n";
    } else {
```

```
    std::cout << "Not all elements in the array are
even.\n";
  }

   return 0;
}
```

99. Write a program that checks if an array contains duplicate elements.

```
#include <iostream>

int main() {
  int n;

   // Get the size of the array
  std::cout << "Enter the number of elements: ";
  std::cin >> n;

   int arr[n];

   // Input elements
  std::cout << "Enter the elements of the array:\n";
  for (int i = 0; i < n; i++) {
```

```
    std::cin >> arr[i];
  }

  // Check for duplicates
  bool hasDuplicates = false;
  for (int i = 0; i < n - 1; i++) {
    for (int j = i + 1; j < n; j++) {
      if (arr[i] == arr[j]) {
        hasDuplicates = true;
        break;
      }
    }
    if (hasDuplicates) break;  // exit outer loop if a duplicate is found
  }

  if (hasDuplicates) {
    std::cout << "The array contains duplicate elements.\n";
  } else {
    std::cout << "The array does not contain duplicate elements.\n";
  }
```

```
    return 0;
}
```

100. Given an array of random numbers, push all the zero's of the array to the end without changing the order of non-zero elements.

```
#include <iostream>
#include <vector>

int main() {
  int n;

    std::cout << "Enter the number of elements: ";
  std::cin >> n;

    std::vector<int> nums(n);

    std::cout << "Enter the elements of the array:\n";
  for (int i = 0; i < n; i++) {
    std::cin >> nums[i];
```

```
    }

    int count = 0;  // Count of non-zero elements

    // Traverse the array and if an element is non-zero, then place the element at next available position
    for (int i = 0; i < n; i++) {
        if (nums[i] != 0) {
            nums[count++] = nums[i];
        }
    }

    // Set remaining positions to zero
    while (count < n) {
        nums[count++] = 0;
    }

    std::cout << "Array after pushing zeros to the end:\n";
    for (int num : nums) {
        std::cout << num << " ";
    }
```

```
    std::cout << std::endl;

    return 0;
}
```

101. Given an unsorted array of integers, find the length of the longest consecutive elements sequence.

```
#include <iostream>
#include <unordered_set>
#include <vector>

int main() {
    std::vector<int> nums;
    std::unordered_set<int> set;
    int n, input, longestStreak = 0;

    std::cout << "Enter the number of elements: ";
    std::cin >> n;

    std::cout << "Enter the elements of the array:\n";
    for (int i = 0; i < n; i++) {
```

```
        std::cin >> input;
        nums.push_back(input);
        set.insert(input);
    }

    for (int num : nums) {
        // Check if this number is the potential start of
a sequence
        if (set.find(num - 1) == set.end()) {
            int currentNum = num;
            int currentStreak = 1;

                    // Check for the next elements in the
sequence
            while (set.find(currentNum + 1) != set.end()) {
                currentNum += 1;
                currentStreak += 1;
            }

                    // Update the longest streak
            if (currentStreak > longestStreak) {
                longestStreak = currentStreak;
            }
```

```
    }
  }

    std::cout << "The length of the longest consecu-
tive elements sequence is: "
        << longestStreak << std::endl;

    return 0;
}
```

102. Given an array of integers, find all pairs in the array that sum up to a specified value.

```
#include <iostream>
#include <unordered_set>
#include <vector>

int main() {
  std::vector<int> nums;
  std::unordered_set<int> set;
  int n, input, longestStreak = 0;
```

```
    std::cout << "Enter the number of elements: ";
   std::cin >> n;

    std::cout << "Enter the elements of the array:\n";
   for (int i = 0; i < n; i++) {
     std::cin >> input;
     nums.push_back(input);
     set.insert(input);
   }

    for (int num : nums) {
     // Check if this number is the potential start of
a sequence
     if (set.find(num - 1) == set.end()) {
       int currentNum = num;
       int currentStreak = 1;

                 // Check for the next elements in the
sequence
       while (set.find(currentNum + 1) != set.end()) {
         currentNum += 1;
         currentStreak += 1;
```

```
        }

                // Update the longest streak
        if (currentStreak > longestStreak) {
          longestStreak = currentStreak;
        }
      }
    }

    std::cout << "The length of the longest consecutive elements sequence is: "
          << longestStreak << std::endl;

    return 0;
}
```

103. Make an algorithm that reads an array of n positions, and then creates a new vector that calculates the difference of each element with the next element.

```
#include <iostream>
```

```cpp
#include <vector>

int main() {
  int n;

   std::cout << "Enter the number of elements: ";
  std::cin >> n;

   if (n < 2) {
    std::cout << "Please enter a value greater than 1 for meaningful differences." << std::endl;
    return 1;
  }

   std::vector<int> nums(n);
  std::vector<int> differences(n - 1);

   std::cout << "Enter the elements of the array:\n";
  for (int i = 0; i < n; i++) {
    std::cin >> nums[i];
  }

   for (int i = 0; i < n - 1; i++) {
    differences[i] = nums[i] - nums[i + 1];
```

```
    }

    std::cout << "The differences between consecu-
tive elements are:\n";
    for (int diff : differences) {
        std::cout << diff << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

104. Given an array of integers, find the subarray of given length with the least average.

```
#include <iostream>
#include <vector>

int main() {
    int n, k;

    std::cout << "Enter the number of elements: ";
```

```
    std::cin >> n;

     std::vector<int> nums(n);

     std::cout << "Enter the elements of the array:\n";
    for (int i = 0; i < n; i++) {
       std::cin >> nums[i];
    }

     std::cout << "Enter the length of the subarray: ";
    std::cin >> k;

     if (k > n) {
       std::cout << "Subarray length is larger than
array size!" << std::endl;
       return 0;
    }

     // Calculate the sum of the first 'k' elements
    int sum = 0;
    for (int i = 0; i < k; i++) {
       sum += nums[i];
    }
```

```
    int min_sum = sum;
    int start_index = 0;  // Starting index of minimum sum subarray

    // Sliding window approach
    for (int i = k; i < n; i++) {
        sum += nums[i] - nums[i - k];
        if (sum < min_sum) {
            min_sum = sum;
            start_index = i - k + 1;
        }
    }

    std::cout << "The subarray of length " << k << " with the least average is:\n";
    for (int i = start_index; i < start_index + k; i++) {
        std::cout << nums[i] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

105. Given two sorted arrays, merge them into a single sorted array.

```
#include <iostream>
#include <vector>

int main() {
  int n1, n2;

    // Read first array
  std::cout << "Enter the number of elements in the first array: ";
  std::cin >> n1;
  std::vector<int> arr1(n1);
  std::cout << "Enter the sorted elements of the first array:\n";
  for (int i = 0; i < n1; i++) {
    std::cin >> arr1[i];
  }

    // Read second array
  std::cout << "Enter the number of elements in the
```

```
second array: ";
    std::cin >> n2;
    std::vector<int> arr2(n2);
    std::cout << "Enter the sorted elements of the
second array:\n";
    for (int i = 0; i < n2; i++) {
        std::cin >> arr2[i];
    }

    // Merging arrays
    int i = 0, j = 0;
    std::vector<int> merged(n1 + n2);
    int k = 0;  // Index for merged array
    while (i < n1 && j < n2) {
        if (arr1[i] <= arr2[j]) {
            merged[k++] = arr1[i++];
        } else {
            merged[k++] = arr2[j++];
        }
    }

    // If there are remaining elements in arr1
    while (i < n1) {
```

```
        merged[k++] = arr1[i++];
    }

    // If there are remaining elements in arr2
    while (j < n2) {
        merged[k++] = arr2[j++];
    }

    // Print merged array
    std::cout << "Merged sorted array is:\n";
    for (int m = 0; m < k; m++) {
        std::cout << merged[m] << " ";
    }
    std::cout << std::endl;

    return 0;
}
```

Strings

106. Create a program that reads two words and concatenates them, displaying the resulting word.

```
#include <iostream>
#include <string>

int main() {
  std::string word1, word2;

   // Read the first word
  std::cout << "Enter the first word: ";
  std::cin >> word1;

   // Read the second word
  std::cout << "Enter the second word: ";
  std::cin >> word2;

   // Concatenate the words
```

```
    std::string concatenatedWord = word1 + word2;

    // Display the resulting word
    std::cout << "Resulting word: " << concatenated-
Word << std::endl;

    return 0;
}
```

107. Write a program that takes a word and displays each letter separately.

```
#include <iostream>
#include <string>

int main() {
    std::string word;

    // Read the word
    std::cout << "Enter a word: ";
    std::cin >> word;

    // Display each letter separately
    for (int i = 0; i < word.length(); i++) {
```

```
        std::cout << word[i] << std::endl;
    }

    return 0;
}
```

108. Create a program that takes a sentence and replaces all the letters "a" with "e".

```
#include <iostream>
#include <string>

int main() {
    std::string sentence;

    // Get the entire line including spaces
    std::cout << "Enter a sentence: ";
    std::getline(std::cin, sentence);

    // Replace every 'a' with 'e'
    for (int i = 0; i < sentence.length(); i++) {
        if (sentence[i] == 'a') {
```

```
            sentence[i] = 'e';
        }
    }

    // Display the modified sentence
    std::cout << "Modified sentence: " << sentence <<
std::endl;

    return 0;
}
```

109. Create a program that takes a sentence and replaces all spaces with a new line.

```
#include <iostream>
#include <string>

int main() {
    std::string sentence;

    // Get the entire line including spaces
    std::cout << "Enter a sentence: ";
```

```
    std::getline(std::cin, sentence);

    // Replace every space with a newline
    for (int i = 0; i < sentence.length(); i++) {
        if (sentence[i] == ' ') {
            sentence[i] = '\n';
        }
    }

    // Display the modified sentence
    std::cout << "Modified sentence:\n" << sentence
<< std::endl;

    return 0;
}
```

110. Write a program that receives a name and checks that it starts with the letter "A".

```
#include <iostream>
#include <string>
```

```
int main() {
   std::string name;

    // Get the name input
   std::cout << "Enter a name: ";
   std::getline(std::cin, name);

    // Check if the first character of the name is 'A'
   if (name[0] == 'A' || name[0] == 'a') {
      std::cout << "The name starts with the letter 'A'.
\n";
   } else {
      std::cout << "The name does not start with the
letter 'A'.\n";
   }

    return 0;
}
```

111. Write a program that reads a word and checks if it is a palindrome (if it can be read backwards the same way).

```
#include <iostream>
#include <string>
#include <algorithm>

int main() {
  std::string word;

    // Get the word input
  std::cout << "Enter a word: ";
  std::cin >> word;

    std::string reversedWord = word;
  std::reverse(reversedWord.begin(), reversed-
Word.end());

    // Check if the word and its reversed version are
the same
  if (word == reversedWord) {
```

```
        std::cout << word << " is a palindrome.\n";
    } else {
        std::cout << word << " is not a palindrome.\n";
    }

    return 0;
}
```

112. Create a program that reads two words and checks if the second word is an anagram of the first.

```
#include <iostream>
#include <string>
#include <algorithm>

int main() {
    std::string word1, word2;

    // Get the word inputs
    std::cout << "Enter the first word: ";
    std::cin >> word1;
```

```
    std::cout << "Enter the second word: ";
    std::cin >> word2;

    // If the lengths are different, they can't be anagrams
    if (word1.length() != word2.length()) {
        std::cout << "The words are not anagrams.\n";
        return 0;
    }

    // Sort both words
    std::sort(word1.begin(), word1.end());
    std::sort(word2.begin(), word2.end());

    // Check if the sorted words are the same
    if (word1 == word2) {
        std::cout << "The words are anagrams.\n";
    } else {
        std::cout << "The words are not anagrams.\n";
    }

    return 0;
}
```

113. Write a program that takes a full name and displays only the first name.

```
#include <iostream>
#include <string>

int main() {
   std::string fullName;

    // Get the full name input
   std::cout << "Enter the full name: ";
   std::getline(std::cin, fullName);

    // Find the first space in the name
   size_t spacePosition = fullName.find(' ');

    std::string firstName;


    // If a space was found, extract the first name
   if (spacePosition != std::string::npos) {
     firstName = fullName.substr(0, spacePosition);
   } else {
```

```
    // If no space was found, assume the full input
is the first name
    firstName = fullName;
  }

  std::cout << "First name: " << firstName <<
std::endl;

  return 0;
}
```

114. Make a program that receives a sentence and displays the amount of blank spaces present in it.

```
#include <iostream>
#include <string>

int main() {
  std::string sentence;
  int count = 0;

  // Get the sentence input
```

```
    std::cout << "Enter a sentence: ";
    std::getline(std::cin, sentence);

    // Loop through each character and count
spaces
    for (char c : sentence) {
        if (c == ' ') {
            count++;
        }
    }

    std::cout << "Number of blank spaces: " << count
<< std::endl;

    return 0;
}
```

115. Create a program that reads a word and displays the number of vowels present in it.

```
#include <iostream>
#include <string>
```

```
int main() {
   std::string word;
   int count = 0;

    // Get the word input
   std::cout << "Enter a word: ";
   std::cin >> word;

    // Loop through each character and count vow-
els
   for (char c : word) {
     // Convert the character to lowercase for uni-
formity
      c = std::tolower(c);
      if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
         count++;
      }
   }

    std::cout << "Number of vowels: " << count <<
std::endl;

    return 0;
```

```
}
```

116. Create a program that reads a word and displays the number of vowels and consonants present in it.

```
#include <iostream>
#include <string>
#include <cctype>

int main() {
   std::string word;
   int vowelCount = 0, consonantCount = 0;

    // Get the word input
   std::cout << "Enter a word: ";
   std::cin >> word;

    // Loop through each character and count vowels and consonants
   for (char c : word) {
     // Convert the character to lowercase for uniformity
```

```
        c = std::tolower(c);
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
            vowelCount++;
        } else if (std::isalpha(c)) { // Ensure the character is a letter
            consonantCount++;
        }
    }

    std::cout << "Number of vowels: " << vowelCount << std::endl;
    std::cout << "Number of consonants: " << consonantCount << std::endl;

    return 0;
}
```

117. Write a program that takes a full name and displays the last name (last name) first.

```
#include <iostream>
```

```cpp
#include <string>
#include <sstream>

int main() {
   std::string fullName, word, lastName;

    std::cout << "Enter your full name: ";
   std::getline(std::cin, fullName);

    std::stringstream ss(fullName);

    // Extract words until the end, keeping the last
word as the potential last name
   while (ss >> word) {
     lastName = word;
   }

    std::cout << "Last name first: " << lastName << ",
" << fullName.substr(0, fullName.rfind(lastName))
<< std::endl;

    return 0;
}
```

118. Implement a method to perform basic string compression using the counts of repeated characters. For instance, the string aabccccccaaa would become a2b1c5a3.

```
#include <iostream>
#include <string>
#include <sstream>

int main() {
   std::string input;
   std::cout << "Enter a string: ";
   std::cin >> input;

    if (input.empty()) {
     std::cout << "Compressed string: " << input <<
std::endl;
     return 0;
   }
```

```
  std::stringstream compressed;
  char currentChar = input[0];
  int count = 1;

  for (int i = 1; i < input.length(); i++) {
   if (input[i] == currentChar) {
     count++;
   } else {
     compressed << currentChar << count;
     currentChar = input[i];
     count = 1;
   }
  }

  // Append the last character and its count
  compressed << currentChar << count;

  std::string result = compressed.str();
  if (result.length() < input.length()) {
    std::cout << "Compressed string: " << result <<
std::endl;
  } else {
```

```
    std::cout << "Compressed string: " << input <<
std::endl;
  }

    return 0;
}
```

119. Rawwords: Write an algorithm that checks whether a word is a "rawword". A word is considered a "prime word" if the sum of the letter values (where 'a' = 1, 'b' = 2, etc.) is a prime number.

```
#include <iostream>
#include <string>

int main() {
  std::string word;
  std::cout << "Enter a word: ";
  std::cin >> word;

    int sum = 0;
```

```
for(char c : word) {
    // Ensure the character is a lowercase letter
    if(c >= 'a' && c <= 'z') {
        sum += (c - 'a' + 1); // 'a' = 1, 'b' = 2, etc.
    }
}

    // Check for prime. A prime number is only
divisible by 1 and itself.
bool isPrime = true;

    // Corner cases: 0 and 1 are not prime numbers
if(sum <= 1) {
    isPrime = false;
}
// Check for divisibility from 2 to sqrt(sum)
for(int i = 2; i*i <= sum && isPrime; i++) {
    if(sum % i == 0) {
        isPrime = false;
    }
}

    if(isPrime) {
```

```
        std::cout << word << " is a rawword." <<
std::endl;
    } else {
        std::cout << word << " is not a rawword." <<
std::endl;
    }

    return 0;
}
```

120. Given a list of strings, write a function to determine the longest common prefix.

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

int main() {
    // Sample list of strings
    std::vector<std::string> strs = {"flow-
```

```
er","flow","flight"};

    if (strs.empty()) {
     std::cout << "No common prefix" << std::endl;
     return 0;
   }

    // Take the shortest string as the potential
prefix
   std::string prefix = *min_element(strs.begin(),
strs.end(), [](const std::string &s1, const std::string
&s2) {
     return s1.size() < s2.size();
   });

    for (const std::string &str : strs) {
     // Compare the current prefix with each string
in the list
     // If a mismatch is found, reduce the size of the
prefix and try again
     while (str.substr(0, prefix.size()) != prefix) {
       prefix = prefix.substr(0, prefix.size() - 1);
       if (prefix.empty()) {
```

```
            std::cout << "No common prefix" <<
std::endl;
            return 0;
        }
    }
  }


    std::cout << "Longest Common Prefix: " << prefix
<< std::endl;

    return 0;
}
```

Matrices

121. Write a program that fills a 3x3 matrix with values entered by the user and displays the sum of the main diagonal values.

```
#include <iostream>

int main() {
   int matrix[3][3];
   int diagonalSum = 0;

    std::cout << "Enter the values for a 3x3 matrix:"
<< std::endl;

    // Reading values into the matrix
   for (int i = 0; i < 3; i++) {
     for (int j = 0; j < 3; j++) {
       std::cout << "Enter value for position [" << i <<
```

```
"][" << j << "]: ";
            std::cin >> matrix[i][j];
        }
    }

    // Calculating the sum of the main diagonal
    for (int i = 0; i < 3; i++) {
        diagonalSum += matrix[i][i];
    }

    std::cout << "The sum of the main diagonal val-
ues is: " << diagonalSum << std::endl;

    return 0;
}
```

122. Write a program that fills a 4x4 matrix with random values and displays the transposed matrix.

```
#include <iostream>
#include <ctime>
#include <cstdlib>
```

```
int main() {
  int matrix[4][4];
  int transposed[4][4];

   // Seed for random number generator
  std::srand(static_cast<unsigned int>(std::
time(nullptr)));

   std::cout << "Original Matrix:" << std::endl;

   // Filling the matrix with random values and
displaying them
  for (int i = 0; i < 4; i++) {
    for (int j = 0; j < 4; j++) {
      matrix[i][j] = std::rand() % 100;  // Filling with
random values between 0 and 99
       std::cout << matrix[i][j] << "\t";
    }
    std::cout << std::endl;
  }

   // Transposing the matrix
  for (int i = 0; i < 4; i++) {
```

```
        for (int j = 0; j < 4; j++) {
            transposed[j][i] = matrix[i][j];
        }
    }

    std::cout << "\nTransposed Matrix:" <<
std::endl;

    // Displaying the transposed matrix
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 4; j++) {
            std::cout << transposed[i][j] << "\t";
        }
        std::cout << std::endl;
    }

    return 0;
}
```

123. Write a program that fills a 5x5 matrix with integers and displays the largest value in the matrix and its position.

```
#include <iostream>

int main() {
   int matrix[5][5];
   int largestValue = INT_MIN;  // Initialize with smallest possible integer
   int rowIndex = 0, colIndex = 0;

    std::cout << "Enter the values for the 5x5 matrix:" << std::endl;

    // Filling the matrix with values entered by the user
   for (int i = 0; i < 5; i++) {
     for (int j = 0; j < 5; j++) {
        std::cout << "Enter value for [" << i << "][" << j << "]: ";
```

```
            std::cin >> matrix[i][j];

                    // Check if the current value is the
largest
            if (matrix[i][j] > largestValue) {
              largestValue = matrix[i][j];
              rowIndex = i;
              colIndex = j;
            }
        }
    }

    // Display the largest value and its position
    std::cout << "The largest value in the matrix is "
<< largestValue
              << " at position [" << rowIndex << "][" << col-
Index << "]" << std::endl;

    return 0;
}
```

124. Count negative numbers in a matrix

```
#include <iostream>

int main() {
  int matrix[5][5];
  int negativeCount = 0;

   std::cout << "Enter the values for the 5x5 ma-
trix:" << std::endl;

   // Filling the matrix with values entered by the
user
  for (int i = 0; i < 5; i++) {
    for (int j = 0; j < 5; j++) {
      std::cout << "Enter value for [" << i << "][" << j
<< "]: ";
      std::cin >> matrix[i][j];

            // Check if the current value is negative
      if (matrix[i][j] < 0) {
        negativeCount++;
      }
```

```
    }
  }

    // Display the count of negative numbers
   std::cout << "The number of negative values in
the matrix is: " << negativeCount << std::endl;

    return 0;
}
```

125. Write a program that reads a 3x3 matrix and calculates the average of the values present in the even positions (sum of the even indices) of the matrix.

```
#include <iostream>

int main() {
   int matrix[3][3];
   int sum = 0;
   int count = 0;

    std::cout << "Enter the values for the 3x3 ma-
```

```
trix:" << std::endl;

    // Filling the matrix with values entered by the
user
  for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 3; j++) {
      std::cout << "Enter value for [" << i << "][" << j
<< "]: ";
      std::cin >> matrix[i][j];

            // Check if both row and column
indices are even
      if (i % 2 == 0 && j % 2 == 0) {
        sum += matrix[i][j];
        count++;
      }
    }
  }

    // Calculate average
  double average = static_cast<double>(sum) /
count;
```

```
    // Display the average of the even positions
   std::cout << "The average of values in the even po-
sitions is: " << average << std::endl;

    return 0;
}
```

126. Calculate the sum of boundary elements of a matrix.

```
#include <iostream>

int main() {
   int M, N;

    // Ask user for dimensions of the matrix
   std::cout << "Enter number of rows (M): ";
   std::cin >> M;

    std::cout << "Enter number of columns (N): ";
   std::cin >> N;

    int matrix[M][N];
```

```
  int boundarySum = 0;

    // Input the matrix elements
  std::cout << "Enter the elements of the matrix:"
<< std::endl;
  for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
      std::cout << "matrix[" << i << "][" << j << "]: ";
      std::cin >> matrix[i][j];


                // Check if the element is on the
boundary and add it to the boundarySum
      if (i == 0 || i == M-1 || j == 0 || j == N-1) {
        boundarySum += matrix[i][j];
      }
    }
  }

    // Display the sum of boundary elements
  std::cout << "The sum of the boundary elements
is: " << boundarySum << std::endl;
```

```
    return 0;
}
```

127. Write a program that reads a 4x4 matrix and checks if it is a diagonal matrix, that is, if all elements outside the main diagonal are equal to zero.

```
#include <iostream>

int main() {
   const int SIZE = 4;
   int matrix[SIZE][SIZE];
   bool isDiagonal = true;

    // Input the matrix elements
   std::cout << "Enter the elements of the 4x4 matrix:" << std::endl;
   for (int i = 0; i < SIZE; i++) {
      for (int j = 0; j < SIZE; j++) {
         std::cout << "matrix[" << i << "][" << j << "]: ";
```

```
            std::cin >> matrix[i][j];
        }
    }

    // Check if the matrix is diagonal
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (i != j && matrix[i][j] != 0) {
                isDiagonal = false;
                break;
            }
        }
        if (!isDiagonal) {
            break;
        }
    }

    if (isDiagonal) {
        std::cout << "The matrix is a diagonal matrix."
<< std::endl;
    } else {
        std::cout << "The matrix is not a diagonal ma-
trix." << std::endl;
```

```
    }

    return 0;
}
```

128. Write a program to determine whether a given matrix is symmetric.

```
#include <iostream>

int main() {
    const int SIZE = 4; // You can change this to any desired size
    int matrix[SIZE][SIZE];

    // Input the matrix elements
    std::cout << "Enter the elements of the " << SIZE << "x" << SIZE << " matrix:" << std::endl;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            std::cout << "matrix[" << i << "][" << j << "]: ";
            std::cin >> matrix[i][j];
        }
```

```
    }

    // Check if the matrix is symmetric
    bool isSymmetric = true;
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (matrix[i][j] != matrix[j][i]) {
                isSymmetric = false;
                break;
            }
        }
        if (!isSymmetric) {
            break;
        }
    }

    if (isSymmetric) {
        std::cout << "The matrix is symmetric." <<
std::endl;
    } else {
        std::cout << "The matrix is not symmetric." <<
std::endl;
    }
```

```
    return 0;
}
```

129. Write a program that fills a 4x4 matrix with random numbers and displays the sum of the values present in each row and in each column.

```
#include <iostream>
#include <ctime>
#include <cstdlib>

int main() {
   const int SIZE = 4;
   int matrix[SIZE][SIZE];
   int rowSum[SIZE] = {0};
   int colSum[SIZE] = {0};

    // Seed random number generator
   srand(time(0));

    // Fill matrix with random numbers and display
```

```
the matrix
   std::cout << "Matrix:\n";
   for (int i = 0; i < SIZE; i++) {
      for (int j = 0; j < SIZE; j++) {
         matrix[i][j] = 1 + rand() % 100; // random
numbers between 1 and 100
         std::cout << matrix[i][j] << "\t";
         rowSum[i] += matrix[i][j];
         colSum[j] += matrix[i][j];
      }
      std::cout << "\n";
   }

    // Display the sum of values in each row
   std::cout << "\nRow sums:\n";
   for (int i = 0; i < SIZE; i++) {
      std::cout << "Row " << i + 1 << ": " << rowSum[i]
<< "\n";
   }

    // Display the sum of values in each column
   std::cout << "\nColumn sums:\n";
   for (int j = 0; j < SIZE; j++) {
```

```
        std::cout << "Column " << j + 1 << ": " << col-
Sum[j] << "\n";
    }

    return 0;
}
```

130. Write a program that reads a 3x3 matrix and calculates the determinant of the matrix.

```
#include <iostream>

int main() {
    const int SIZE = 3;
    double matrix[SIZE][SIZE];

    // Read the matrix
    std::cout << "Enter the elements of the 3x3 ma-
trix:\n";
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            std::cout << "Element [" << i + 1 << "][" << j + 1
```

```
<< "]: ";
            std::cin >> matrix[i][j];
        }
    }

    // Calculate determinant
    double determinant = 0.0;
    determinant += matrix[0][0] * (matrix[1][1] *
matrix[2][2] - matrix[1][2] * matrix[2][1]);
    determinant -= matrix[0][1] * (matrix[1][0] * ma-
trix[2][2] - matrix[1][2] * matrix[2][0]);
    determinant += matrix[0][2] * (matrix[1][0] * ma-
trix[2][1] - matrix[1][1] * matrix[2][0]);

    // Display the determinant
    std::cout << "The determinant of the matrix is: "
<< determinant << "\n";

    return 0;
}
```

131. Write a program that reads two 2x2 matrices and displays the sum of the two matrices.

```
#include <iostream>

int main() {
   const int SIZE = 2;
   double matrix1[SIZE][SIZE], matrix2[SIZE][SIZE],
result[SIZE][SIZE];

    // Read the first matrix
   std::cout << "Enter the elements of the first 2x2
matrix:\n";
   for (int i = 0; i < SIZE; i++) {
      for (int j = 0; j < SIZE; j++) {
         std::cout << "Element [" << i + 1 << "][" << j + 1
<< "]: ";
         std::cin >> matrix1[i][j];
      }
   }
```

```
    // Read the second matrix
    std::cout << "Enter the elements of the second
2x2 matrix:\n";
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            std::cout << "Element [" << i + 1 << "][" << j + 1
<< "]: ";
            std::cin >> matrix2[i][j];
        }
    }

    // Calculate the sum
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            result[i][j] = matrix1[i][j] + matrix2[i][j];
        }
    }

    // Display the result
    std::cout << "Sum of the two matrices is:\n";
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
```

```
            std::cout << result[i][j] << " ";
        }
        std::cout << "\n";
    }

    return 0;
}
```

132. Write a program that reads two matrices and returns the multiplication between them as an answer. The program should observe whether or not it is possible to perform the multiplication between the two matrices.

```
#include <iostream>
#include <vector>

int main() {
    int rows1, cols1, rows2, cols2;

    // Get the dimensions of the first matrix
```

```
    std::cout << "Enter the number of rows and columns for the first matrix: ";
    std::cin >> rows1 >> cols1;

    // Read the first matrix
    std::vector<std::vector<double>> matrix1(rows1, std::vector<double>(cols1));
    std::cout << "Enter elements of the first matrix:\n";
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols1; j++) {
            std::cin >> matrix1[i][j];
        }
    }

    // Get the dimensions of the second matrix
    std::cout << "Enter the number of rows and columns for the second matrix: ";
    std::cin >> rows2 >> cols2;

    // Read the second matrix
    std::vector<std::vector<double>> matrix2(rows2, std::vector<double>(cols2));
```

```
    std::cout << "Enter elements of the second matrix:\n";
    for (int i = 0; i < rows2; i++) {
        for (int j = 0; j < cols2; j++) {
            std::cin >> matrix2[i][j];
        }
    }

    // Check for multiplication possibility
    if (cols1 != rows2) {
        std::cout << "Matrix multiplication is not possible.";
        return 0;
    }

    // Perform multiplication
    std::vector<std::vector<double>> result(rows1, std::vector<double>(cols2, 0));
    for (int i = 0; i < rows1; i++) {
        for (int j = 0; j < cols2; j++) {
            for (int k = 0; k < cols1; k++) {
                result[i][j] += matrix1[i][k] * matrix2[k][j];
            }
```

```
    }
  }

  // Display the result
  std::cout << "Result of matrix multiplication:\n";
  for (int i = 0; i < rows1; i++) {
    for (int j = 0; j < cols2; j++) {
      std::cout << result[i][j] << " ";
    }
    std::cout << "\n";
  }

  return 0;
}
```

133. Print a matrix in spiral order.

```
#include <iostream>
#include <vector>

int main() {
  int rows, cols;

  // Read matrix dimensions
```

```
  std::cout << "Enter the number of rows and columns: ";
  std::cin >> rows >> cols;

   // Read matrix values
  std::vector<std::vector<int>> matrix(rows, std::vector<int>(cols));
  std::cout << "Enter matrix elements:\n";
  for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
      std::cin >> matrix[i][j];
    }
  }

   int top = 0, bottom = rows - 1, left = 0, right = cols - 1;

   // Print matrix in spiral order
  while (top <= bottom && left <= right) {
    // Print top row
    for (int i = left; i <= right; i++) {
      std::cout << matrix[top][i] << " ";
    }
```

```
    top++;

    // Print rightmost column
    for (int i = top; i <= bottom; i++) {
      std::cout << matrix[i][right] << " ";
    }
    right--;

    // Print bottom row (only if there's more
than one row left to process)
    if (top <= bottom) {
      for (int i = right; i >= left; i--) {
        std::cout << matrix[bottom][i] << " ";
      }
      bottom--;
    }

    // Print leftmost column (only if there's more
than one column left to process)
    if (left <= right) {
      for (int i = bottom; i >= top; i--) {
        std::cout << matrix[i][left] << " ";
      }
```

```
        left++;
      }
    }

    return 0;
}
```

134. Write a program that reads an m x n matrix, indicating the location where there are mines in a Minesweeper game (being 0 for a neutral field, and 1 for locations where there would be mines), and the program should return a matrix indicating, for each position, the number of mines in neighboring houses.

```
#include <iostream>
#include <vector>

int main() {
  int m, n;
```

```
    // Read matrix dimensions
   std::cout << "Enter the number of rows and col-
umns (m n): ";
   std::cin >> m >> n;

    // Read matrix values
   std::vector<std::vector<int>> matrix(m, std::vec-
tor<int>(n));
   std::cout << "Enter matrix elements (0 for neu-
tral, 1 for mine):\n";
   for (int i = 0; i < m; i++) {
      for (int j = 0; j < n; j++) {
         std::cin >> matrix[i][j];
      }
   }

    std::vector<std::vector<int>> result(m, std::vec-
tor<int>(n, 0));

    for (int i = 0; i < m; i++) {
     for (int j = 0; j < n; j++) {
       // If there's a mine at this location, skip count-
```

```
ing for it
        if (matrix[i][j] == 1) continue;

                // Check the surrounding 8 cells
        for (int x = -1; x <= 1; x++) {
          for (int y = -1; y <= 1; y++) {
            if (x == 0 && y == 0) continue;  // Skip the
center cell
            int new_i = i + x;
            int new_j = j + y;

                        if (new_i >= 0 && new_i < m
&& new_j >= 0 && new_j < n && matrix[new_i]
[new_j] == 1) {
              result[i][j]++;
            }
          }
        }
      }
    }

     // Print resulting matrix
    std::cout << "Resulting matrix:\n";
```

```
    for (int i = 0; i < m; i++) {
        for (int j = 0; j < n; j++) {
            if (matrix[i][j] == 1) std::cout << "M ";
            else std::cout << result[i][j] << " ";
        }
        std::cout << "\n";
    }

    return 0;
}
```

135. Make a function that receives a 3x3 matrix representing the game of tic-tac-toe, and check if there is a winner, if there is a tie, or if the game is not over yet

```
#include <iostream>
#include <vector>

enum class GameState {
    X_Wins,
    O_Wins,
```

```
    Tie,
    Ongoing
};

GameState checkTicTacToe(std::vector<std::vec-
tor<char>>& board) {
    // Check rows, columns, diagonals
    for (int i = 0; i < 3; ++i) {
        if (board[i][0] == board[i][1] && board[i][1] ==
board[i][2] && board[i][0] != ' ') {
            return board[i][0] == 'X' ? GameState::X_Wins :
GameState::O_Wins;
        }

         if (board[0][i] == board[1][i] && board[1][i] ==
board[2][i] && board[0][i] != ' ') {
            return board[0][i] == 'X' ? GameState::X_Wins :
GameState::O_Wins;
        }
    }

     // Check diagonals
    if (board[0][0] == board[1][1] && board[1][1] ==
```

```
board[2][2] && board[0][0] != ' ') {
    return board[0][0] == 'X' ? GameState::X_Wins :
GameState::O_Wins;
  }

   if (board[0][2] == board[1][1] && board[1][1] ==
board[2][0] && board[0][2] != ' ') {
    return board[0][2] == 'X' ? GameState::X_Wins :
GameState::O_Wins;
  }

   // Check for a tie or ongoing game
  bool isFull = true;
  for (int i = 0; i < 3 && isFull; ++i) {
    for (int j = 0; j < 3; ++j) {
      if (board[i][j] == ' ') {
        isFull = false;
        break;
      }
    }
  }

   return isFull ? GameState::Tie : GameState::On-
```

```
going;
}

int main() {
  std::vector<std::vector<char>> board(3, std::vec-
tor<char>(3));


   std::cout << "Enter Tic-Tac-Toe board (X, O, or
space for each cell):\n";
  for (int i = 0; i < 3; ++i) {
    for (int j = 0; j < 3; ++j) {
      std::cin >> board[i][j];
    }
  }

   GameState result = checkTicTacToe(board);
  switch (result) {
    case GameState::X_Wins:
      std::cout << "X Wins!\n";
      break;
    case GameState::O_Wins:
      std::cout << "O Wins!\n";
```

```
            break;
        case GameState::Tie:
            std::cout << "It's a Tie!\n";
            break;
        case GameState::Ongoing:
            std::cout << "The game is still ongoing.\n";
            break;
    }

    return 0;
}
```

Recursive Functions

136. Write a recursive function to calculate the factorial of a number.

```
#include <iostream>

long long factorial(int n) {
   if (n <= 1) return 1;
   return n * factorial(n - 1);
}

int main() {
   int num;
   std::cout << "Enter a number: ";
   std::cin >> num;
   std::cout << "Factorial of " << num << " is " << fac-
torial(num) << std::endl;

    return 0;
}
```

137. Implement a recursive function to calculate the Fibonacci sequence up to a given number.

```
#include <iostream>

int fibonacci(int n) {
  if (n <= 1) return n;
  return fibonacci(n - 1) + fibonacci(n - 2);
}

int main() {
  int num;
  std::cout << "Enter a number: ";
  std::cin >> num;


   std::cout << "Fibonacci sequence up to " << num
<< " terms:" << std::endl;
  for (int i = 0; i < num; i++) {
    std::cout << fibonacci(i) << " ";
  }
```

```
    std::cout << std::endl;

    return 0;
}
```

138. Create a recursive function to check if a number is prime.

```
#include <iostream>
#include <cmath>

bool isPrimeRecursive(int n, int i = 2) {
    // Base case: if n is less than 2, it's not prime
    if (n <= 1) return false;


    // If we reach a divisor other than 1 and itself, it's not prime
    if (n % i == 0) return false;


    // If we've checked up to the square root of n and found no divisors, it's prime
```

```
    if (i * i > n) return true;


    // Continue checking the next potential divisor
    return isPrimeRecursive(n, i + 1);
}

int main() {
    int num;
    std::cout << "Enter a number: ";
    std::cin >> num;


    if (isPrimeRecursive(num)) {
        std::cout << num << " is a prime number." <<
std::endl;
    } else {
        std::cout << num << " is not a prime number."
<< std::endl;
    }

    return 0;
}
```

139. Develop a recursive function to calculate the sum of the digits of an integer.

```
#include <iostream>

int sumOfDigits(int n) {
   // Base case: if n is a single digit number, return the number itself
   if (n < 10) return n;


    // Get the last digit and add it to the recursive result of the rest of the number
   return n % 10 + sumOfDigits(n / 10);
}

int main() {
   int num;
   std::cout << "Enter a number: ";
   std::cin >> num;
```

```
    int result = sumOfDigits(num);
   std::cout << "Sum of digits of " << num << " is: " <<
result << std::endl;

    return 0;
}
```

140. Write a recursive function to calculate the power of an integer raised to an exponent.

```
#include <iostream>

int power(int base, int exponent) {
   // Base case: if the exponent is 0, the result is 1
   if (exponent == 0) return 1;


    // Recursive case: multiply the base by the re-
sult of the power function with reduced exponent
   return base * power(base, exponent - 1);
}
```

```
int main() {
   int base, exponent;
   std::cout << "Enter base: ";
   std::cin >> base;
   std::cout << "Enter exponent: ";
   std::cin >> exponent;


    int result = power(base, exponent);
   std::cout << base << " raised to the power " << ex-
ponent << " is: " << result << std::endl;

    return 0;
}
```

141. Implement a recursive function to find the greatest common divisor (GCD) of two numbers.

```
#include <iostream>

int gcd(int a, int b) {
```

```
    // Base case: If b is 0, then a is the GCD
    if (b == 0) return a;

    // Recursive case: Find GCD of b and the remainder when a is divided by b
    return gcd(b, a % b);
}

int main() {
    int num1, num2;
    std::cout << "Enter first number: ";
    std::cin >> num1;
    std::cout << "Enter second number: ";
    std::cin >> num2;


    int result = gcd(num1, num2);
    std::cout << "GCD of " << num1 << " and " << num2 << " is: " << result << std::endl;

    return 0;
}
```

142. Create a recursive function to reverse a string.

```
#include <iostream>
#include <string>

std::string reverseString(const std::string &str) {
  // Base case: If the string is empty or contains just one character
  if (str.length() <= 1) return str;

   // Recursive case: Reverse the substring without the first character
  // and concatenate the first character to the end
  return reverseString(str.substr(1)) + str[0];
}

int main() {
  std::string input;
  std::cout << "Enter a string: ";
  std::cin >> input;
```

```
    std::string reversed = reverseString(input);
    std::cout << "Reversed string: " << reversed <<
std::endl;

    return 0;
}
```

143. Develop a recursive function to find the smallest value in an array.

```
#include <iostream>

int findSmallest(int arr[], int n, int index = 0, int
smallest = INT_MAX) {
    // Base case: if index is equal to n, return the
smallest found so far
    if (index == n) return smallest;

    // If the current element is smaller than the
current smallest,
    // update the smallest value
    if (arr[index] < smallest) smallest = arr[index];
```

```
    // Move to the next element
    return findSmallest(arr, n, index + 1, smallest);
}

int main() {
    int n;
    std::cout << "Enter the number of elements: ";
    std::cin >> n;


    int arr[n];
    std::cout << "Enter " << n << " numbers:" <<
std::endl;
    for (int i = 0; i < n; i++) {
        std::cin >> arr[i];
    }

    int smallest = findSmallest(arr, n);
    std::cout << "Smallest value: " << smallest <<
std::endl;

    return 0;
}
```

144. Write a recursive function to determine whether a word is a palindrome.

```
#include <iostream>
#include <string>

bool isPalindrome(const std::string& word, int start, int end) {
  // Base case: if start is greater than or equal to end, return true (the word is a palindrome)
  if (start >= end) return true;

   // If characters at current indices don't match, return false
  if (word[start] != word[end]) return false;

   // Move to the next set of characters
  return isPalindrome(word, start + 1, end - 1);
}

int main() {
```

```
    std::string word;
    std::cout << "Enter a word: ";
    std::cin >> word;

     if (isPalindrome(word, 0, word.length() - 1)) {
      std::cout << word << " is a palindrome." <<
std::endl;
    } else {
       std::cout << word << " is not a palindrome." <<
std::endl;
    }

     return 0;
}
```

145. Implement a recursive function to calculate the sum of elements of an array.

```
#include <iostream>

int sumArray(int arr[], int size) {
    // Base case: if the size is zero, return 0
    if (size == 0) return 0;
```

```
    // Recursively calculate the sum of the remaining elements and add the current element
    return arr[size - 1] + sumArray(arr, size - 1);
}

int main() {
    int n;
    std::cout << "Enter the number of elements: ";
    std::cin >> n;

     int *arr = new int[n];
    std::cout << "Enter the elements of the array: ";
    for(int i = 0; i < n; i++) {
       std::cin >> arr[i];
    }

     std::cout << "Sum of the array elements: " << sumArray(arr, n) << std::endl;


     delete[] arr; // Don't forget to free the dynamically allocated memory
    return 0;
```

}

Regular Expressions

146. Write a regex to validate email addresses, ensuring it contains @ and . symbols.

```
#include <iostream>
#include <regex>
#include <string>

int main() {
   std::string email;
   std::cout << "Enter email address: ";
   std::cin >> email;

    // Regex pattern to check if email contains @ and . symbols
   std::regex pattern(R"(^[^@]+@[^@]+\.[^@]+$)");

    if (std::regex_match(email, pattern)) {
     std::cout << "Valid email address!" << std::endl;
```

```
    } else {
        std::cout << "Invalid email address!" <<
std::endl;
    }

    return 0;
}
```

147. Extract all valid dates from a string in the format YYYY-MM-DD or DD/MM/YYYY.

```
#include <iostream>
#include <regex>
#include <string>

int main() {
    std::string input;
    std::cout << "Enter the string: ";
    std::getline(std::cin, input);

    // Regex pattern for YYYY-MM-DD and DD/MM/
YYYY date formats
```

```
    std::regex pattern(R"((\d{4}-\d{2}-\d{2})|(\d{2}/\d{2}/\d{4}))");


    std::smatch matches;
    while (std::regex_search(input, matches, pattern)) {
        std::cout << matches[0] << std::endl;
        input = matches.suffix().str();
    }

    return 0;
}
```

148. Check if a given string is a valid credit card number (typically 16 digits, sometimes separated by - or spaces).

```
#include <iostream>
#include <regex>
#include <string>

int main() {
```

```
    std::string input;
    std::cout << "Enter the credit card number: ";
    std::getline(std::cin, input);

     // Regex pattern for a credit card number for-
mat
    std::regex pattern(R"(^(\d{4}-\d{4}-\d{4}-\d{4}|
\d{4} \d{4} \d{4} \d{4}|\d{16})$)");


     if (std::regex_match(input, pattern)) {
      std::cout << "Valid credit card number format!"
<< std::endl;
    } else {
      std::cout << "Invalid credit card number for-
mat!" << std::endl;
    }

     return 0;
}
```

149. Validate passwords to ensure they contain at least one uppercase character, one lowercase character, one digit, one special character, and are at least 8 characters long.

```
#include <iostream>
#include <regex>
#include <string>

int main() {
   std::string input;
   std::cout << "Enter the password: ";
   std::getline(std::cin, input);

     // Regex pattern for password validation
   std::regex pattern(R"((?=.*[a-z])(?=.*[A-Z])(?=.*\d)
(?=.*[\W_]).{8,})");


     if (std::regex_search(input, pattern)) {
```

```
        std::cout << "Valid password!" << std::endl;
    } else {
        std::cout << "Invalid password!" << std::endl;
    }

    return 0;
}
```

150. Replace multiple spaces in a text with a single space.

```
#include <iostream>
#include <regex>
#include <string>

int main() {
    std::string input;
    std::cout << "Enter the text: ";
    std::getline(std::cin, input);

    std::regex pattern(R"(\s+)");
    std::string replacement(" ");
```

```
    std::string result = std::regex_replace(input,
pattern, replacement);

    std::cout << "Processed text: " << result <<
std::endl;

    return 0;
}
```

151. Validate and extract currency values from a text (e.g., $100.25).

```
#include <iostream>
#include <regex>
#include <string>

int main() {
   std::string input;
   std::cout << "Enter the text: ";
   std::getline(std::cin, input);

    std::regex pattern(R"((\$\d+(\.\d{1,2})?))");
```

```
    std::smatch matches;
  std::string::const_iterator searchStart(input.cbe-
gin());
  while (std::regex_search(searchStart, input.
cend(), matches, pattern)) {
    std::cout << "Found currency value: " <<
matches[0] << '\n';
    searchStart = matches.suffix().first;
  }

    return 0;
}
```

152. Write a regex to extract all URLs from a text.

```
#include <iostream>
#include <regex>
#include <string>

int main() {
  std::string input;
```

```
    std::cout << "Enter the text: ";
    std::getline(std::cin, input);

     std::regex pattern(R"(\bhttps?://[^\s<>"]+|www
\.[^\s<>"]+)");


     std::smatch matches;
    std::string::const_iterator searchStart(input.cbe-
gin());
    while (std::regex_search(searchStart, input.
cend(), matches, pattern)) {
        std::cout << "Found URL: " << matches[0] << '\n';
        searchStart = matches.suffix().first;
    }

     return 0;
}
```

153. Extract all hex color codes from a text (e.g., #AABBCC or #ABC).

```
#include <iostream>
```

```
#include <regex>
#include <string>

int main() {
   std::string input;
   std::cout << "Enter the text: ";
   std::getline(std::cin, input);

    std::regex pattern(R"(#([A-Fa-f0-9]{6}|[A-Fa-
f0-9]{3}))");


    std::smatch matches;
   std::string::const_iterator searchStart(input.cbe-
gin());
   while (std::regex_search(searchStart, input.
cend(), matches, pattern)) {
      std::cout << "Found hex color: " << matches[0]
<< '\n';
      searchStart = matches.suffix().first;
   }

    return 0;
```

```
}
```

154. Extract all valid IPv4 addresses from a text.

```
#include <iostream>
#include <regex>
#include <string>

int main() {
   std::string input;
   std::cout << "Enter the text: ";
   std::getline(std::cin, input);

     std::regex pattern(R"(\b(25[0-5]|2[0-4][0-9]|
[01]?[0-9][0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9]
[0-9]?)\.(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.
(25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\b)");


     std::smatch matches;
   std::string::const_iterator searchStart(input.cbe-
gin());
```

```
    while (std::regex_search(searchStart, input.
cend(), matches, pattern)) {
        std::cout << "Found IPv4 address: " <<
matches[0] << '\n';
        searchStart = matches.suffix().first;
    }

    return 0;
}
```

155. Use capture groups to rearrange date formats (convert YYYY-MM-DD to DD-MM-YYYY).

```
#include <iostream>
#include <regex>
#include <string>

int main() {
    std::string input;
    std::cout << "Enter the date in format YYYY-MM-
DD: ";
```

```
    std::getline(std::cin, input);

    std::regex pattern(R"((\d{4})-(\d{2})-(\d{2}))");
    std::string replacement("$3-$2-$1");


    std::string newFormat = std::regex_replace(input, pattern, replacement);
    std::cout << "Reformatted date: " << newFormat << '\n';

    return 0;
}
```

Sorting Algorithms

156. Implement the Bubble Sort algorithm

```
#include <iostream>

void bubbleSort(int arr[], int n) {
  for (int i = 0; i < n-1; i++) {       // Loop over each
element
    for (int j = 0; j < n-i-1; j++) {  // Loop over the
unsorted part
      if (arr[j] > arr[j+1]) {      // Compare the adja-
cent elements
        // Swap arr[j] and arr[j+1]
        int temp = arr[j];
        arr[j] = arr[j+1];
        arr[j+1] = temp;
      }
    }
  }
```

```
}

int main() {
   int arr[] = {64, 34, 25, 12, 22, 11, 90};
   int n = sizeof(arr) / sizeof(arr[0]);

    std::cout << "Original array: \n";
   for (int i = 0; i < n; i++)
      std::cout << arr[i] << " ";
   std::cout << "\n";

    bubbleSort(arr, n);

    std::cout << "Sorted array: \n";
   for (int i = 0; i < n; i++)
      std::cout << arr[i] << " ";
   std::cout << "\n";

    return 0;
}
```

157. Implement the Selection Sort algorithm

```
#include <iostream>

void selectionSort(int arr[], int n) {
  for (int i = 0; i < n-1; i++) {
    int min_idx = i;              // Find the minimum element in the unsorted array
    for (int j = i+1; j < n; j++) {
      if (arr[j] < arr[min_idx]) {
        min_idx = j;
      }
    }

    // Swap the found minimum element with the first element
    int temp = arr[min_idx];
    arr[min_idx] = arr[i];
    arr[i] = temp;
  }
```

```
}

int main() {
   int arr[] = {64, 25, 12, 22, 11};
   int n = sizeof(arr) / sizeof(arr[0]);

    std::cout << "Original array: \n";
   for (int i = 0; i < n; i++)
      std::cout << arr[i] << " ";
   std::cout << "\n";

    selectionSort(arr, n);

    std::cout << "Sorted array: \n";
   for (int i = 0; i < n; i++)
      std::cout << arr[i] << " ";
   std::cout << "\n";

    return 0;
}
```

158. Implement the Insertion Sort algorithm

```
#include <iostream>

void insertionSort(int arr[], int n) {
  for (int i = 1; i < n; i++) {
    int key = arr[i];
    int j = i - 1;

      // Move elements of arr[0..i-1] that are greater than the key
    // to one position ahead of their current position
    while (j >= 0 && arr[j] > key) {
      arr[j + 1] = arr[j];
      j = j - 1;
    }
    arr[j + 1] = key;
  }
}
```

```
int main() {
   int arr[] = {12, 11, 13, 5, 6};
   int n = sizeof(arr) / sizeof(arr[0]);

    std::cout << "Original array: \n";
   for (int i = 0; i < n; i++)
     std::cout << arr[i] << " ";
   std::cout << "\n";

    insertionSort(arr, n);

    std::cout << "Sorted array: \n";
   for (int i = 0; i < n; i++)
     std::cout << arr[i] << " ";
   std::cout << "\n";

    return 0;
}
```

159. Implement the Merge Sort algorithm

```
#include <iostream>
```

```
void merge(int arr[], int l, int m, int r) {
  int n1 = m - l + 1;
  int n2 = r - m;

   // Create temp arrays
  int L[n1], R[n2];

   // Copy data to temp arrays L[] and R[]
  for (int i = 0; i < n1; i++)
    L[i] = arr[l + i];
  for (int j = 0; j < n2; j++)
    R[j] = arr[m + 1 + j];

   int i = 0;
  int j = 0;
  int k = l;

   // Merge the temp arrays back into arr[l..r]
  while (i < n1 && j < n2) {
    if (L[i] <= R[j]) {
      arr[k] = L[i];
      i++;
    } else {
```

```
            arr[k] = R[j];
            j++;
        }
        k++;
    }

    // Copy the remaining elements of L[], if there
are any
    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    // Copy the remaining elements of R[], if there
are any
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
```

```
void mergeSort(int arr[], int l, int r) {
  if (l >= r)
    return; // Returns recursively

   int m = l + (r - l) / 2;
  mergeSort(arr, l, m);
  mergeSort(arr, m + 1, r);
  merge(arr, l, m, r);
}

int main() {
  int arr[] = {12, 11, 13, 5, 6, 7};
  int n = sizeof(arr) / sizeof(arr[0]);

   std::cout << "Given array: \n";
  for (int i = 0; i < n; i++)
    std::cout << arr[i] << " ";
  std::cout << "\n";

   mergeSort(arr, 0, n - 1);

   std::cout << "Sorted array: \n";
  for (int i = 0; i < n; i++)
```

```
        std::cout << arr[i] << " ";
    std::cout << "\n";

    return 0;
}
```

160. Implement the Quick Sort algorithm

```
#include <iostream>

// Function to swap two elements
void swap(int* a, int* b) {
    int t = *a;
    *a = *b;
    *b = t;
}

// Function that partitions the array using the last
element as the pivot
int partition(int arr[], int low, int high) {
    int pivot = arr[high];
    int i = (low - 1);

    for (int j = low; j <= high - 1; j++) {
```

```
    if (arr[j] < pivot) {
      i++;
      swap(&arr[i], &arr[j]);
    }
  }
  swap(&arr[i + 1], &arr[high]);
  return (i + 1);
}

void quickSort(int arr[], int low, int high) {
  if (low < high) {
    // pi is partitioning index
    int pi = partition(arr, low, high);

      // Separately sort elements before and after
partition
    quickSort(arr, low, pi - 1);
    quickSort(arr, pi + 1, high);
  }
}

int main() {
  int arr[] = {10, 7, 8, 9, 1, 5};
```

```
    int n = sizeof(arr) / sizeof(arr[0]);

     std::cout << "Given array: \n";
    for (int i = 0; i < n; i++)
       std::cout << arr[i] << " ";
    std::cout << "\n";

     quickSort(arr, 0, n - 1);

     std::cout << "Sorted array: \n";
    for (int i = 0; i < n; i++)
       std::cout << arr[i] << " ";
    std::cout << "\n";

     return 0;
}
```

Complete List of Exercises

Basic Exercises

1. Write a program that prints "Hello World!" on the screen.

2. Write a program that prints "Hello" on one line and "World!" on the bottom line, using two different commands.

3. Write a program that prints "Hello" on one line and "World!" on the bottom line, using only one command to print.

4. Write a program that prints "Hello" on one line and "World!" on the bottom line, leaving a blank line between the words.

5. Write a program that prints "Hello World!" (in the same line) on the screen, using one command to print "Hello" and another command to print "World!".

6. Write a program that prints "Hello World!" (in the same line) on the screen, using one command to print "Hello", another command to print "World", and another to print "!".

7. Make a program that asks for the user's name, and prints the name on the screen immediately after.

8. Make a program that reads the names of three people in sequence, followed by their respective ages, and prints each name with their respective ages on the screen.

9. Make a program that reads the names of three people in sequence, followed by their respective ages and heights, and prints the data in table

format on the screen.

10. Make a program that reads the names of three people in sequence, followed by reading their respective ages and heights, and prints the data on the screen in table format, using dashes (-) to separate rows and pipes (|) to separate columns.

Mathematical Formulas

11. Write a program that prompts the user for two numbers and displays the addition, subtraction, multiplication, and division between them.

12. Write a program that calculates the arithmetic mean of two numbers entered by the user.

13. Create a program that calculates and displays the arithmetic mean of three numbers entered by the user.

14. Write a program that reads two numbers, the first being the base and the second the exponent, and then printing the first number raised to the second.

15. Write a program that reads a number and prints the square root of the number on the screen.

16. Write a program that calculates the geometric mean of three numbers entered by the user → GeoMean=3n1n2n3

17. Write a program that calculates the BMI of an individual, using the formula BMI = weight / height²

18. Create a program that calculates and displays the perimeter of a circle, prompting the user for the radius, using the formula $P = 2\pi r$

19. Write a program that calculates the area of a circle from the radius, using the formula $A = \pi r^2$

20. Write a program that calculates the delta of a quadratic equation ($\Delta = b^2 - 4ac$).

21. Write a program that calculates the perimeter and area of a rectangle, using the formulas $P = 2(w + l)$ and $A = wl$, where w is the width and l is the length

22. Write a program that calculates the perimeter and area of a triangle, using the formulas $P = a + b + c$ and $A = (b * h) / 2$, where a, b and c are the sides of the triangle and h is the height relative to the side B.

23. Write a program that calculates the average velocity of an object, using the formula $v = \Delta s / \Delta t$, where v is the average velocity, Δs is the space variation, and Δt is the time variation

24. Write a program that calculates the kinetic energy of a moving object, using the formula $E = (mv^2) / 2$, where E is the kinetic energy, m is

the mass of the object, and v is the velocity.

25. Write a program that calculates the work done by a force acting on an object, using the formula W = F * d, where W is the work, F is the applied force, and d is the distance traveled by the object.

26. Write a program that calculates the n-th term of an arithmetic progression given the value of the first term, the common difference (ratio), and the value of n read from the user → an=a 1+(n-1)r

27. Write a program that reads the x and y position of two points in the Cartesian plane, and calculates the distance between them

28. Create a program that prompts the user for the radius of a sphere and calculates and displays its volume.

29. Make a program that reads the resistance

of two resistors and displays the resulting resistance value when connected in series, adding the values, and in parallel using the formula (R1 * R2) / (R1 + R2)

30. Make a program that reads a value of a product and the tax (in percentage), and calculates the final value of it.

Conditionals

31. Make a program that asks for a person's age and displays whether they are of legal age or not (age >= 18).

32. Write a program that reads a number and reports whether it is odd or even.

33. Write a program that reads a number and reports whether it is positive, negative or zero.

34. Create a program that reads three numbers and checks if their sum is positive, negative or equal to zero

35. Write a program that reads two numbers and tells you which one is bigger.

36. Write a program that asks the user for three numbers and displays the largest one.

37. Make a program that reads three numbers, and informs if their sum is divisible by 5 or not.

38. Write a program that asks for an integer and checks if it is divisible by 3 and 5 at the same time.

39. Make a program that asks for two numbers and displays if the first is divisible by the second

40. Make a program that reads the scores of two tests and reports whether the student passed (score greater than or equal to 6) or failed (score less than 6) in each of the tests.

41. Make a program that reads the grades of two tests, calculates the simple arithmetic mean, and informs whether the student passed

(average greater than or equal to 6) or failed (average less than 6).

42. Make a program that reads the age of three people and how many of them are of legal age (age 18 or older).

43. Make a program that reads three numbers, and displays them on the screen in ascending order.

44. Write a program that reads three numbers and tells you if they can be the sides of a triangle (the sum of two sides must always be greater than the third side).

45. Make a program that reads the year of birth of a person and informs if he is able to vote (age greater than or equal to 16 years old).

46. Create a program that asks for a person's age and displays whether they are a child (0-12 years old), teenager (13-17 years old),

adult (18-59 years old), or elderly (60 years old or older), using nested conditionals, without using logical operators.

47. Create a program that asks for a person's age and displays whether they are a child (0-12 years old), teenager (13-17 years old), adult (18-59 years old), or elderly (60 years old or older), using logical operators, without using else, elif, etc.

48. Make a program that reads a person's age and informs if he is not able to vote (age less than 16 years old), if he is able to vote but is not obligated (16, 17 years old, or age equal to or greater than 70 years), or if it is obligatory (18 to 69 years old).

49. Make a program that reads three grades from a student and reports whether he passed (final grade greater than or equal to 7), failed

(final grade less than 4) or was in recovery (final grade between 4 and 7).

50. Write a program that asks for a person's height and weight and calculates their body mass index (BMI), displaying the corresponding category (underweight, normal weight, overweight, obese, severely obese).

51. Write a program that asks the user to enter their grade (0-100) and prints their letter grade (A for 90-100, B for 80-89, C for 70-79, D for 60-69, F for under 60).

52. Write a program that reads the values a, b and c of a quadratic equation, and says if the roots of the function are real or imaginary.

53. Create a program that reads the price of a product. If the price is more than US$200, the tax is 5%, if it is more than US$500, the tax is 7.5%. Products up to US$200 do not pay

taxes. Print the final price of the product on the screen.

54. Write a program that reads a year (a four digit number) and checks whether it is a leap year or not.

55. Input three numbers representing the coefficients of a quadratic equation (a, b, c) and find its roots, also mention if they are real or complex

Repeat Loops

56. Write a program that displays the numbers 1 through 10 using a while loop.

57. Write a program that displays the numbers 1 through 10 using a for loop.

58. Write a program that displays all numbers from 1 to 100, using a while loop.

59. Write a program that displays all numbers from 1 to 100, using a for loop.

60. Write a program that displays all numbers from 1 to 100, using a do-while loop.

61. Write a program that prints all even numbers from 1 to 100 using a while loop and an if statement inside the loop to check if the number is even or not.

62. Write a program that prints all even numbers from 1 to 100 using a while loop without using any if statement.

63. Write a program that displays even numbers 1 to 50 and odd numbers 51 to 100 using a repeating loop.

64. Create a program that prompts the user for a number and displays the table of that number using a loop.

65. Create a program that displays the table of all numbers from 1 to 10.

66. Write a program that calculates and

displays the value of the power of a number entered by the user raised to an exponent also entered by the user, using repetition loops.

67. Create a program that displays the first N first perfect squares, where N is informed by the user, using a loop.

68. Write a program that asks the user for a number N and says whether it is prime or not.

69. Write a program that prompts the user for a number N and displays all prime numbers less than N.

70. Create a program that displays the first N prime numbers, where N is informed by the user, using a loop.

71. Write a program that prompts the user for two numbers A and B and displays all numbers between A and B.

72. Write a program that reads numbers from

the user until a negative number is entered, and prints the sum of the positive numbers.

73. Write a program that asks the user for a number N and displays the sum of all numbers from 1 to N.

74. Write a program that calculates and displays the sum of even numbers from 1 to 100 using a repeating loop.

75. Write a program that prompts the user for a number and displays the Fibonacci sequence up to the given number using a repeating loop.

76. Write a program that reads numbers from the user until zero is entered, and displays the average of the numbers entered.

77. Write a program that prompts the user for a list of numbers, until the user types the number zero, and displays the largest and smallest numbers in the list.

78. Write a program that prompts the user for a number and displays its divisors.

79. Write a program that determines the greatest common divisor (GCD) between two numbers entered by the user.

80. Write a program that determines the lowest common multiple (LCM) between two numbers entered by the user.

81. Write a program that calculates the series below up to the tenth element:

82. Rewrite the previous exercise code until the difference between the terms is less than 0.001.

83. Make a program that calculates the value of sine using the Taylor series according to the equation below until the difference between the terms is less than 0.001.

84. Make a program that calculates the value of cosine using the Taylor series according to the

equation below until the difference between the terms is less than 0.001.

85. Write a program that displays the sine and cosine value of all numbers from 0 to 6.3, with a step of 0.1, using Taylor series to calculate the respective sines and cosines.

Arrays

86. Write a program that reads an array of integers and displays the elements in reverse order.

87. Create a program that reads an array of integers and displays the sum of all the elements.

88. Write a program that reads an array of integers and displays the average of the elements.

89. Write a program that reads an array of integers and displays the largest element in the

array.

90. Create a program that reads two vectors of integers of the same size and displays a new vector with the sum of the corresponding elements of the two vectors.

91. Write a program that reads two arrays of integers with the same size and displays a new array with the elements resulting from the multiplication of the corresponding elements of the two arrays.

92. Write a program that reads an array of integers and displays how many times a specific number appears in the array.

93. Write a program that reads an array of integers and checks if they are in ascending order.

94. Create a program that reads an array of integers and finds the second largest element in

the array.

95. Given an array, find the kth largest and smallest element in it.

96. Write a program that checks if an array contains duplicate elements.

97. Given an array of size n, find an element that appears more than n/2 times. (This element is called the majority element.)

98. Create a program that reads an array of integers and checks if all elements are even.

99. Write a program that checks if an array contains duplicate elements.

100. Given an array of random numbers, push all the zero's of the array to the end without changing the order of non-zero elements.

101. Given an unsorted array of integers, find the length of the longest consecutive elements sequence.

102. Given an array of integers, find all pairs in the array that sum up to a specified value.

103. Make an algorithm that reads an array of n positions, and then creates a new vector that calculates the difference of each element with the next element.

104. Given an array of integers, find the subarray of given length with the least average.

105. Given two sorted arrays, merge them into a single sorted array.

Strings

106. Create a program that reads two words and concatenates them, displaying the resulting word.

107. Write a program that takes a word and displays each letter separately.

108. Create a program that takes a sentence and replaces all the letters "a" with "e".

109. Create a program that takes a sentence and replaces all spaces with a new line.

110. Write a program that receives a name and checks that it starts with the letter "A".

111. Write a program that reads a word and checks if it is a palindrome (if it can be read backwards the same way).

112. Create a program that reads two words and checks if the second word is an anagram of the first.

113. Write a program that takes a full name and displays only the first name.

114. Make a program that receives a sentence and displays the amount of blank spaces present in it.

115. Create a program that reads a word and displays the number of vowels present in it.

116. Create a program that reads a word and

displays the number of vowels and consonants present in it.

117. Write a program that takes a full name and displays the last name (last name) first.

118. Implement a method to perform basic string compression using the counts of repeated characters. For instance, the string aabcccccaaa would become a2b1c5a3.

119. Rawwords: Write an algorithm that checks whether a word is a "rawword". A word is considered a "prime word" if the sum of the letter values (where 'a' = 1, 'b' = 2, etc.) is a prime number.

120. Given a list of strings, write a function to determine the longest common prefix.

Matrices

121. Write a program that fills a 3x3 matrix with values entered by the user and displays the

sum of the main diagonal values.

122. Write a program that fills a 4x4 matrix with random values and displays the transposed matrix.

123. Write a program that fills a 5x5 matrix with integers and displays the largest value in the matrix and its position.

124. Count negative numbers in a matrix

125. Write a program that reads a 3x3 matrix and calculates the average of the values present in the even positions (sum of the even indices) of the matrix.

126. Calculate the sum of boundary elements of a matrix.

127. Write a program that reads a 4x4 matrix and checks if it is a diagonal matrix, that is, if all elements outside the main diagonal are equal to zero.

128. Write a program to determine whether a given matrix is symmetric.

129. Write a program that fills a 4x4 matrix with random numbers and displays the sum of the values present in each row and in each column.

130. Write a program that reads a 3x3 matrix and calculates the determinant of the matrix.

131. Write a program that reads two 2x2 matrices and displays the sum of the two matrices.

132. Write a program that reads two matrices and returns the multiplication between them as an answer. The program should observe whether or not it is possible to perform the multiplication between the two matrices.

133. Print a matrix in spiral order.

134. Write a program that reads an m x n

matrix, indicating the location where there are mines in a Minesweeper game (being 0 for a neutral field, and 1 for locations where there would be mines), and the program should return a matrix indicating, for each position, the number of mines in neighboring houses.

135. Make a function that receives a 3x3 matrix representing the game of tic-tac-toe, and check if there is a winner, if there is a tie, or if the game is not over yet

Recursive Functions

136. Write a recursive function to calculate the factorial of a number.

137. Implement a recursive function to calculate the Fibonacci sequence up to a given number.

138. Create a recursive function to check if a number is prime.

139. Develop a recursive function to calculate the sum of the digits of an integer.

140. Write a recursive function to calculate the power of an integer raised to an exponent.

141. Implement a recursive function to find the greatest common divisor (GCD) of two numbers.

142. Create a recursive function to reverse a string.

143. Develop a recursive function to find the smallest value in an array.

144. Write a recursive function to determine whether a word is a palindrome.

145. Implement a recursive function to calculate the sum of elements of an array.

Regular Expressions

146. Write a regex to validate email addresses, ensuring it contains @ and . symbols.

147. Extract all valid dates from a string in the format YYYY-MM-DD or DD/MM/YYYY.

148. Check if a given string is a valid credit card number (typically 16 digits, sometimes separated by - or spaces).

149. Validate passwords to ensure they contain at least one uppercase character, one lowercase character, one digit, one special character, and are at least 8 characters long.

150. Replace multiple spaces in a text with a single space.

151. Validate and extract currency values from a text (e.g., $100.25).

152. Write a regex to extract all URLs from a text.

153. Extract all hex color codes from a text (e.g., #AABBCC or #ABC).

154. Extract all valid IPv4 addresses from a text.

155. Use capture groups to rearrange date formats (convert YYYY-MM-DD to DD-MM-YYYY).

Sorting Algorithms

156. Implement the Bubble Sort algorithm

157. Implement the Selection Sort algorithm

158. Implement the Insertion Sort algorithm

159. Implement the Merge Sort algorithm

160. Implement the Quick Sort algorithm

Additional Content

In case you want to access the code of all the exercises, you can get it from the link below:

https://forms.gle/FbhMBdTEfoC3zsSt5

Each file is named with the exercise number, with the extension .cpp

About the Author

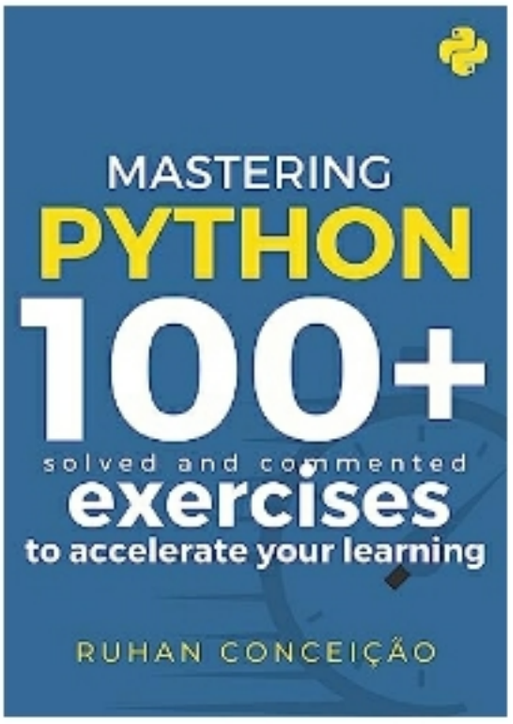
Ruhan Avila da Conceição (@ruhanconceicao, on social media) holds a degree in Computer Engineering (2015) from the Federal University of Pelotas and a Master's in Computing (2016) also from the same university. Since 2018 he has been a professor at the Federal Institute of Education, Science and Technology in the field of Informatics, where he teaches, among others, the disciplines of Object Oriented Programming, Visual Programming and Mobile Device Programming.

In 2014, even as an undergraduate student, Ruhan received the title of Researcher from Rio Grande do Sul in the Young Innovator category due to his academic and scientific career during his years at the university. His research topic has always been related to the algorithmic development of efficient solutions for encoding videos in real time. Still with regard to scientific research, Ruhan accumulates dozens of works published in national and international congresses, as well as articles in scientific journals of great relevance, and two computer programs registered at the National Institute of Intellectual Property.

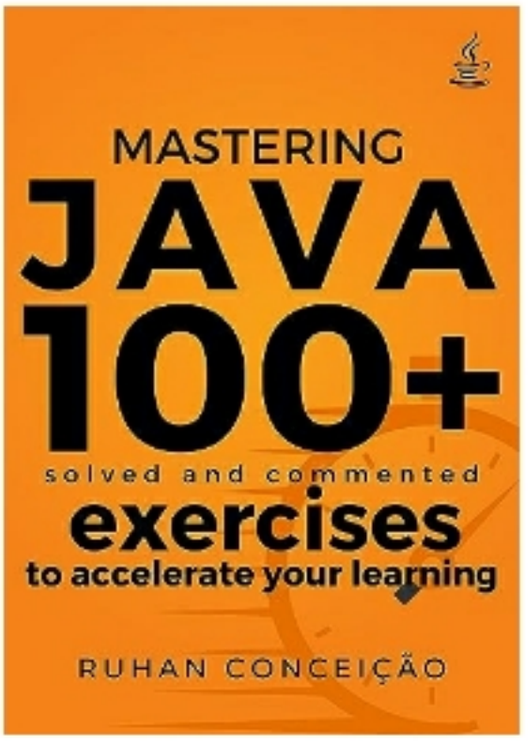
Initiated in programming in the C language, Ruhan Conceição has extensive knowledge in JavaScript languages, as well as their libraries and frameworks ReactJS, React Native, NextJS;

and Java. In addition, the author has also developed projects in Python, C#, Matlab, Google-Script and C++.

Mastering Python

	***Mastering Python:*** *100+ Solved and Commented Exercises to Accelerate Your Learning* https://a.co/d/bGRbAoE

Mastering Java

	***Mastering Java:*** *100+ Solved and Commented Exercises to Accelerate Your Learning* https://a.co/d/0w0Hcs2