# LAB 7: EUCLIDEAN ALGORITHM

**OBJECTIVE:**

i. To study and understand Euclidean algorithm

ii. To write a C Program to perform Euclidean algorithm

**THEORY:**

This algorithm is named after the Greek mathematician Euclid. If we can show that the common divisors of a and b are the same as the common divisors

of b and r, we will have shown that gcd(a, b) = gcd(b, r), because both pairs must have the same greatest common divisor.

ALGORITHM The Euclidean Algorithm.

**procedure** gcd(a, b: positive integers)

x := a

y := b

**while** y 0

r := x mod y

x := y

y := r

**return** x{gcd(a, b) is x}

**OBSERVATION:**

Euclid’s algorithm was implemented in the C programming language which gave me a better understanding of it and its applications in computer science.

**DISCUSSION:**

This lab work solidified my understanding of Euclid’s algorithm, and I found out how it is used in finding the greatest common divisor of two numbers using computers.

**CONCLUSION:**

This lab helped me better understand how to calculate the greatest common divisor of two numbers using Euclid’s algorithm.

# LAB 8: EXTENDED EUCLIDEAN ALGORITHM

**OBJECTIVE:**

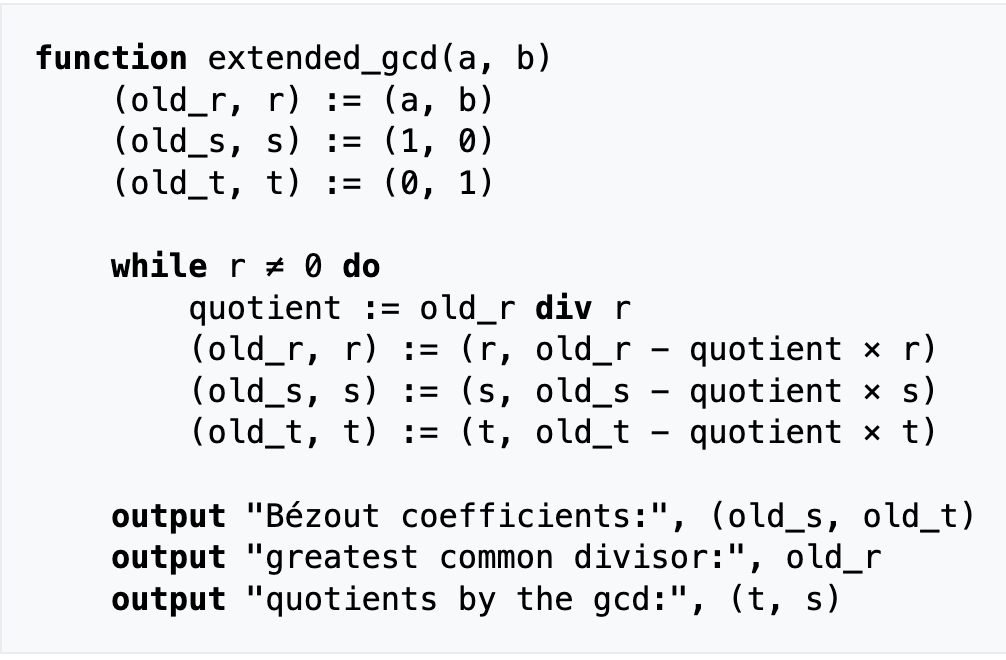
i. To study and understand Extended Euclidean algorithm

ii. To write a C Program to perform Extended Euclidean algorithm

**THEORY:**

This algorithm helps us helps us write the greatest common divisors of two numbers as a linear combination of those two numbers. This is using Bezout’s theorem.

Algorithm:



**OBSERVATION:**

Extended Euclid’s algorithm was implemented in the C programming language which gave me a better understanding of it and its applications in computer science.

**DISCUSSION:**

This lab work solidified my understanding of Extended Euclid’s algorithm, and I found out how it is used in expressing the greatest common divisor of two numbers as a linear combination of those two numbers.

**CONCLUSION:**

This lab helped me better understand how to express the gcd of two numbers as a linear combination of those numbers.

# LAB 9: JOIN AND MEET

**OBJECTIVE:**

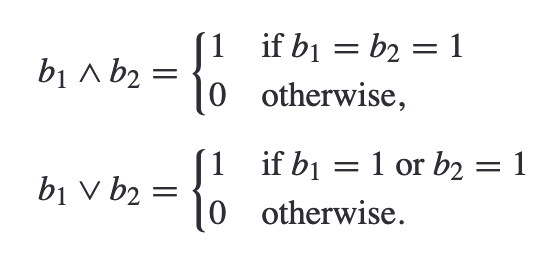
i. To study and understand join and meet operations

ii. To write a C Program to perform join and meet operations

**THEORY:**

Let A = [aij ] and B = [bij ] be m × n zero–one matrices. Then the join of A and B is the zero–one matrix with (i, j )th entry aij ∨ b ij . The join of A and B is denoted by A ∨ B.

The meet of A and B is the zero–one matrix with (i, j )th entry a ij ∧ b ij . The meet of A and B is denoted by A ∧ B.



**OBSERVATION:**

The join and meet operation in Boolean matrices were programmed in the C programming language which made me better understand these operations.

**DISCUSSION:**

This lab work solidified my understanding of join and meet operations, and I found out how it is used in computer science.

**CONCLUSION:**

This lab helped me better understand how to perform join and meet operations in Boolean matrices and how they relate to computer science.

# LAB 10: FACTORIAL USING RECURSION

**OBJECTIVE:**

i. To study and understand recursion

ii. To write a C Program to calculate factorial using recursion

**THEORY:**

Recursion is the process a procedure goes through when one of the steps of the procedure involves invoking the procedure itself. On each iteration, a smaller version of that process is executed until some condition is met which terminates the looping.

The factorial of a number is the product of that number and all the positive integers smaller than that number.

It is denoted by .

**OBSERVATION:**

The factorial operation was implemented using the C programming language using recursion which helped me better understand recursion and how it is applied in programming.

**DISCUSSION:**

This lab work solidified my understanding of recursion and how they are applied in programming to solve problems.

**CONCLUSION:**

This lab helped me better understand how to perform recursion in the C programming language and gave me a good foundation for further programming.

# LAB 11: TRUTH TABLES

**OBJECTIVE:**

i. To study and understand truth tables

ii. To write a C Program to generate a truth table

**THEORY:**

Truth tables are ways of representing logical relationships between Boolean variables. The truth table implemented here calculates the truth values for negation, implication, bi-implication, conjunction, disjunction and exclusive OR.

**OBSERVATION:**

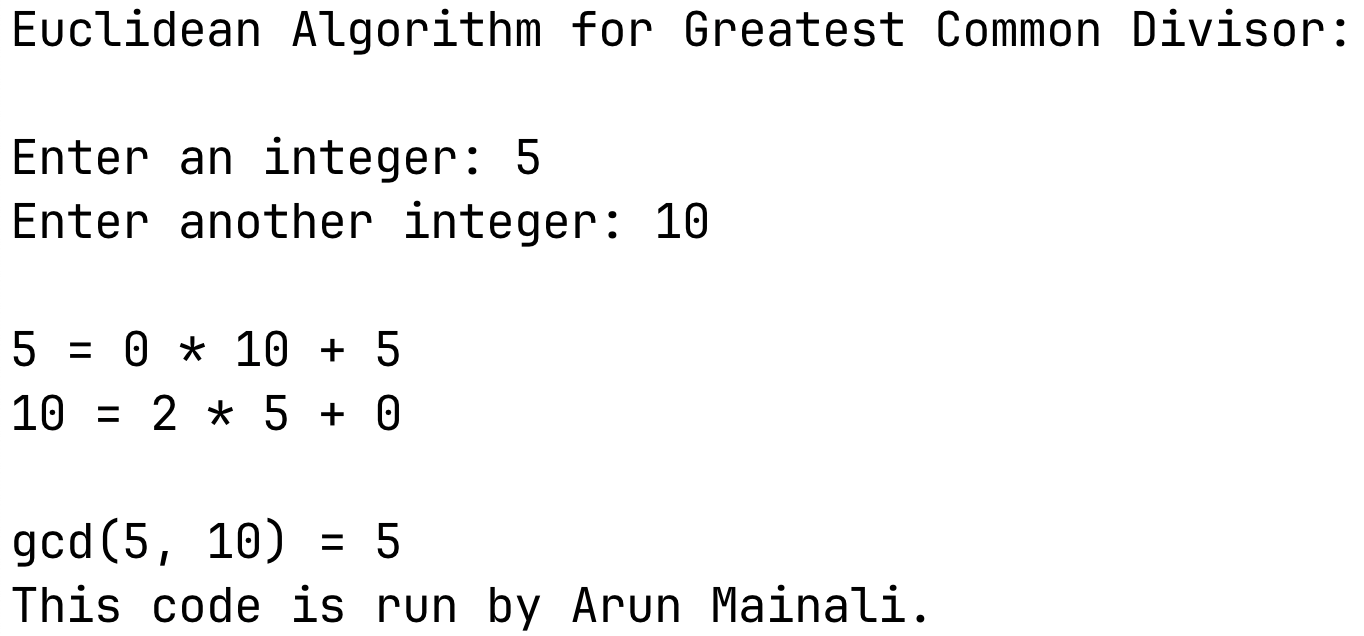
Truth tables were generated using the C programming language and the output was displayed in a readable format. This helped me understand the different logical relationships and truth values of different logical operations.

**DISCUSSION:**

This lab work solidified my understanding of truth tables and how they are applied in programming to solve problems.

**CONCLUSION:**

This lab helped me better understand how to generate truth tables in the C programming language and helped me better understand the different logical relations in a program.

A number and numbers on a white background

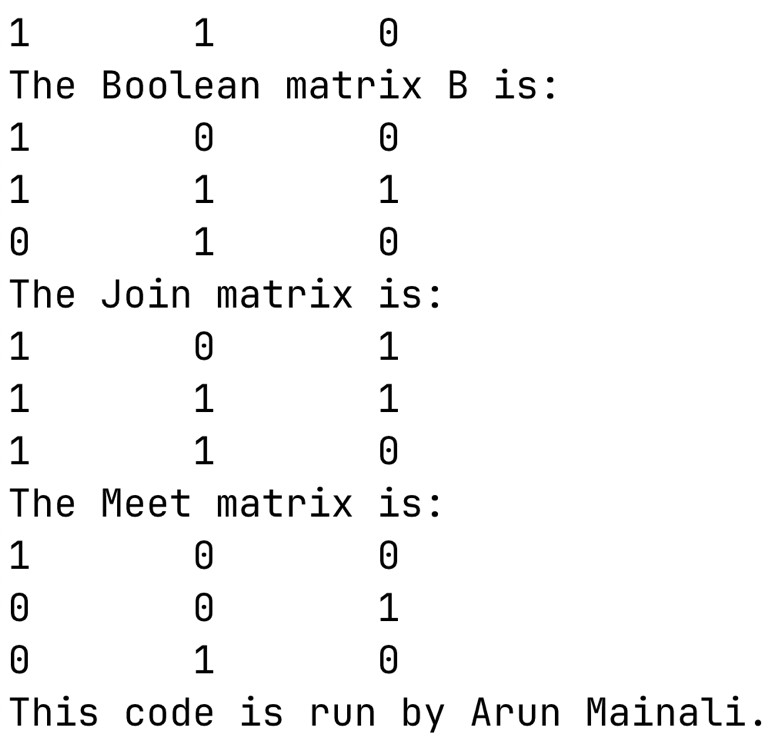
Description automatically generated

A black text with a white background

Description automatically generated

A close-up of a white background

Description automatically generated

A screenshot of a computer

Description automatically generated