**Module Code : FC723**

**Class/Group: Group A**

**Module Title: Programming Theory**

**Assessment Title: Portfolio Project 1**

**Tutor Name: Sophie Norman**

**Student GUID Number: P453118**

**Date of Submission: 27th April 2025**

“I confirm that this assignment is my own work.

Where I/we have referred to academic sources, I have provided in-text citations and included the sources in the final reference list. “

P453118/ FC723/ Sophie Norman

**Pseudocode**

Start

Input: initial\_A

Input: initial\_B

A=initial\_A

B=initial\_B

While B not equal to 0

R=A%B #use modulus to find the remainder

A=B # the new A is the old B

B=R # the new B is the remainder

Result=A # the GCD is the final value of A when B becomes 0

Output: GCD (initial\_A, initial\_B) =Result

End

P453118/ FC723/ Sophie Norman

**Pseudocode(fix)**

CLASS EuclideanAlgorithm

Private\_attribute\_a: integer

Private\_attribute\_b: integer

CONSTRUCTOR (a: integer, b: integer)

FUNCTION get\_gcd() RETURNS integer

While B not equal to 0

R=A%B #use modulus to find the remainder

A=B # the new A is the old B

B=R # the new B is the remainder

Return A

END FUNTION

END Class

input\_a= positive integer

input\_b= positive integer

example=EuclideanAlgorithm(input\_a, input\_b)

gcd\_result=example.get\_gcd()

OUTPUT= "GCD(" + input\_a + ", " + input\_b + ")=" + gcd\_result

P453118/ FC723/ Sophie Norman

Github: <https://github.com/JingNing524/Euclidean>

From my early committed (B2B3FD3) to my last commit (D87376B), I developed and debug with a Euclidian algorithm-based GCD calculator. Here is a step-by-step explanation of the process, which I have included all the GIT commands I used.

Before starting, I ran: git log. This confirmed that my Git user information (name and email) was properly set, and that the repository was tracking commits correctly.

I set up the repository and saved the initial version of my code:

git init: Initialize a new Git repository

git add . : Stage all files

git commit -m " Initial commit: Implemented basic Euclidean Algorithm": Commit the base version of the program.

I refactored the code for reusability and made the input system accept any non-negative integers: git commit -m " Initial finish refactor the code"

I received an error: "Invalid input: Please enter a single number without spaces."

To solve this, I updated the input validation logic: git commit -m "Fix error: "Invalid input: Please enter a single number without spaces.""

Next, I encountered: "'EuclideanAlgorithm' object has no attribute 'valid'"

I corrected the attribute from “self.valid “to “self.\_valid”: git commit -m "Fix the error:'EuclideanAlgorithm' object has no attribute 'valid'"

Once the main logic was stable, I added comments to explain the purpose of each part of the code: git commit -m " Add code comment"

I discovered that entering 0 wasn't allowed due to a condition requiring n > 0. I modified the validation to allow zero as long as both inputs aren't zero git commit -m " fix the problem that i can't input 0" and handled the edge case where the second number is 0, returning the first number as the GCD. git commit -m "fix second input problem"

P453118/ FC723/ Sophie Norman

In the final review, I fixed a typo (caculate\_gcd → calculate\_gcd), moved a logic block outside the loop where it belonged, and thoroughly tested the program with a variety of inputs. git commit -m "Check for final details that need improvement"

I used the following command to compare the code before and after: git diff b2b3fd3 d87376b. This helped me visualize all the improvements and better understand how the code evolved.

After finishing development and testing, I linked the local repository to a GitHub repo: git remote add origin <https://github.com/JingNing524/Euclidean>, git branch -M main, git push -u origin main. This uploaded the project to my public GitHub repository and completed my development cycle.

P453118/ FC723/ Sophie Norman

**Pseudocode of Extending the Euclidian Algorithm to Polynomial**

START

CLASS PolynomialEuclideanAlgorithm:

FUNTION \_\_init\_\_(self, poly1, poly2):

If poly1 and poly2 are both zero polynomials:

self.\_valid = False

self.\_poly1 = None

self.\_poly2 = None

Else:

self.\_valid = True

self.\_poly1 = poly1

self.\_poly2 = poly2

self.\_gcd = None

FUNTION is\_valid(self):

Return self.\_valid

FUNTION calculate\_gcd(self):

If not self.\_valid:

Return None

A = self.\_poly1

B = self.\_poly2

While B is not the zero polynomial:

R = A mod B

A = B

B = R

self.\_gcd = Normalize(A)

OUTPUT= self.\_gcd

FUNTION get\_gcd(self):

If not self.\_valid:

Return None

If self.\_gcd is None:

Return self.calculate\_gcd()

Return self.\_gcd

FUNTION get\_valid\_polynomial\_input(prompt):

Loop:

user\_input = Get input string from user

Try:

Parse input as a polynomial expression

Return the parsed polynomial

Catch parse error:

OUTPUT="Invalid polynomial. Please try again."

Main:

OUTPUT= "Welcome to the Polynomial Euclidean Algorithm GCD Calculator"

Loop:

P = get\_valid\_polynomial\_input("Enter first polynomial: ")

Q = get\_valid\_polynomial\_input("Enter second polynomial: ")

solver = PolynomialEuclideanAlgorithm(P, Q)

If solver.is\_valid():

gcd\_poly = solver.get\_gcd()

OUTPUT="GCD(P, Q) = ", gcd\_poly

Break

Else:

OUTPUT="Invalid polynomials entered. Please try again."

END