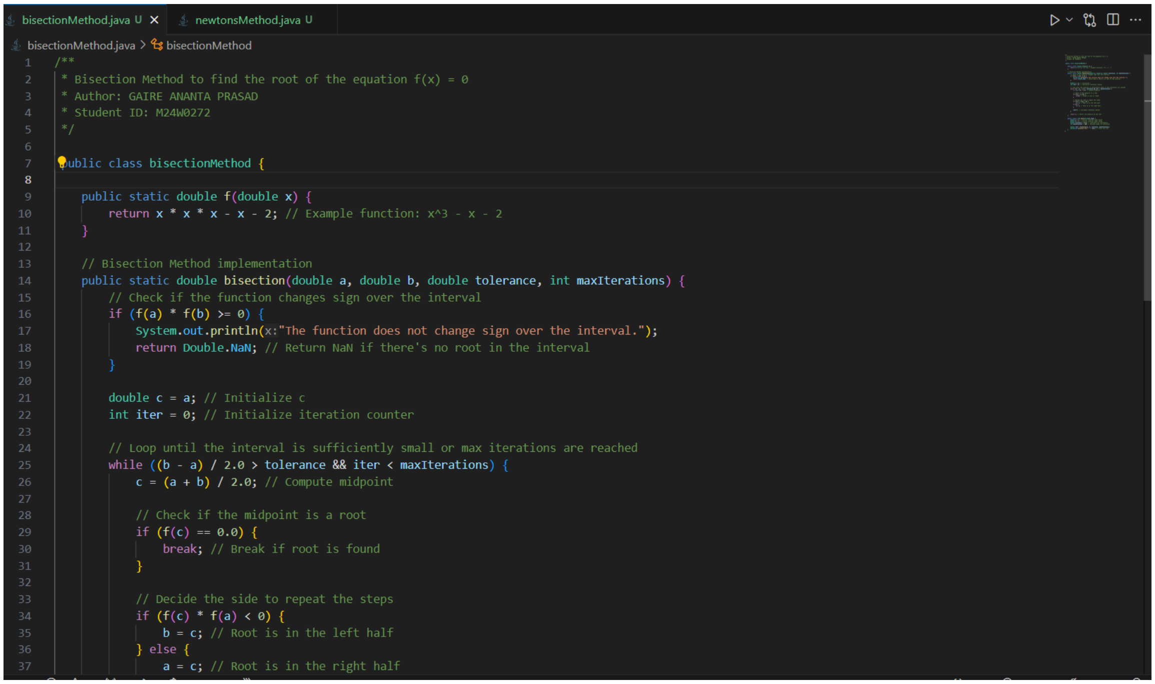
NAME: GAIRE ANANTA PRASAD

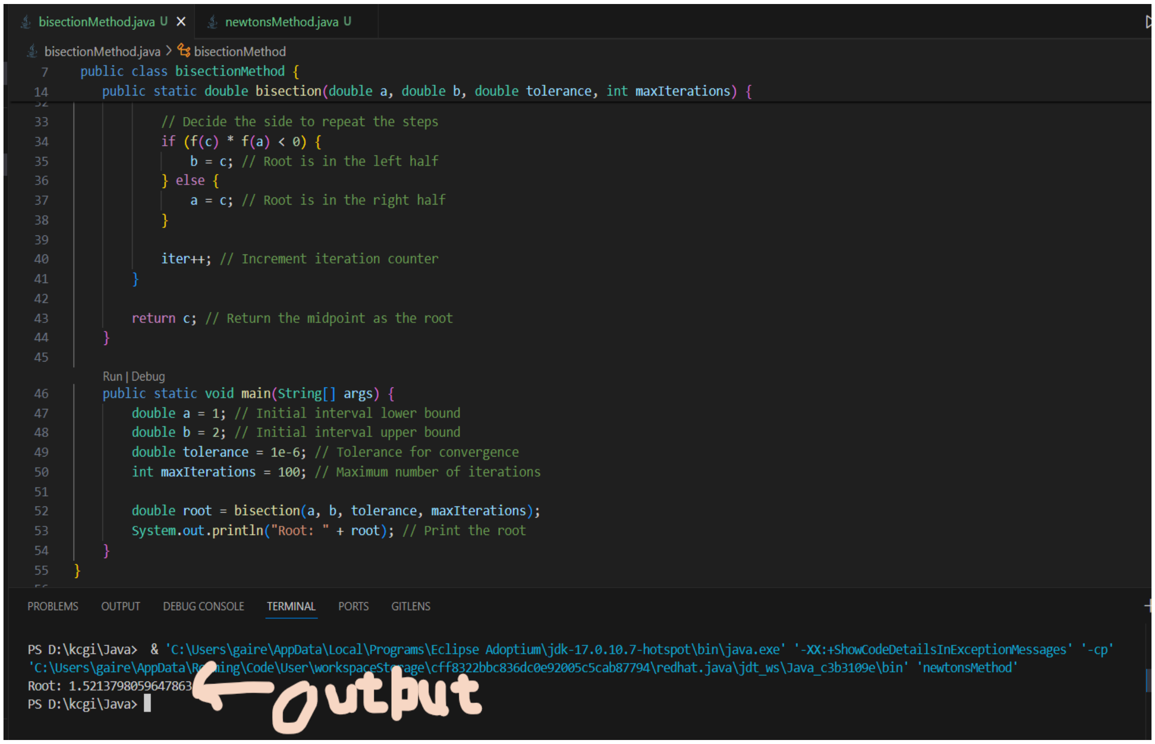
STUDENT ID: M24W0272

1. BISECTION METHOD

   A. CODE FOR BISECTION METHOD



```java
/**
 * Bisection Method to find the root of the equation f(x) = 0
 * Author: GAIRE ANANTA PRASAD
 * Student ID: M24W0272
 */

public class bisectionMethod {

    public static double f(double x) {
        return x * x * x - x - 2; // Example function: x^3 - x - 2
    }

    // Bisection Method implementation
    public static double bisection(double a, double b, double tolerance, int maxIterations) {
        // Check if the function changes sign over the interval
        if (f(a) * f(b) >= 0) {
            System.out.println(x:"The function does not change sign over the interval.");
            return Double.NaN; // Return NaN if there's no root in the interval
        }

        double c = a; // Initialize c
        int iter = 0; // Initialize iteration counter

        // Loop until the interval is sufficiently small or max iterations are reached
        while ((b - a) / 2.0 > tolerance && iter < maxIterations) {
            c = (a + b) / 2.0; // Compute midpoint

            // Check if the midpoint is a root
            if (f(c) == 0.0) {
                break; // Break if root is found
            }

            // Decide the side to repeat the steps
            if (f(c) * f(a) < 0) {
                b = c; // Root is in the left half
            } else {
                a = c; // Root is in the right half
```

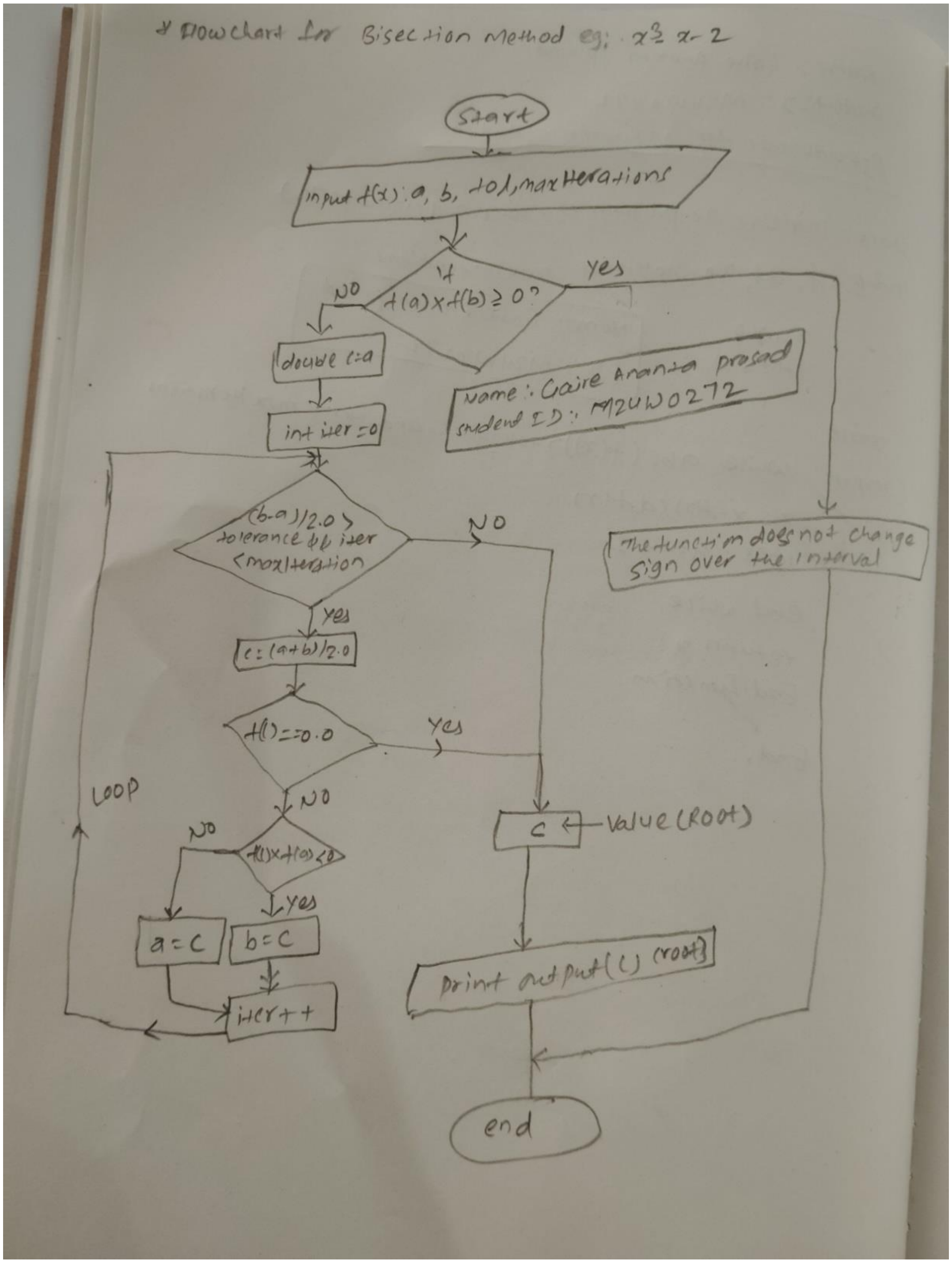


```java
public class bisectionMethod {
    public static double bisection(double a, double b, double tolerance, int maxIterations) {
            // Decide the side to repeat the steps
            if (f(c) * f(a) < 0) {
                b = c; // Root is in the left half
            } else {
                a = c; // Root is in the right half
            }

            iter++; // Increment iteration counter
        }

        return c; // Return the midpoint as the root
    }

    public static void main(String[] args) {
        double a = 1; // Initial interval lower bound
        double b = 2; // Initial interval upper bound
        double tolerance = 1e-6; // Tolerance for convergence
        int maxIterations = 100; // Maximum number of iterations

        double root = bisection(a, b, tolerance, maxIterations);
        System.out.println("Root: " + root); // Print the root
    }
}
```

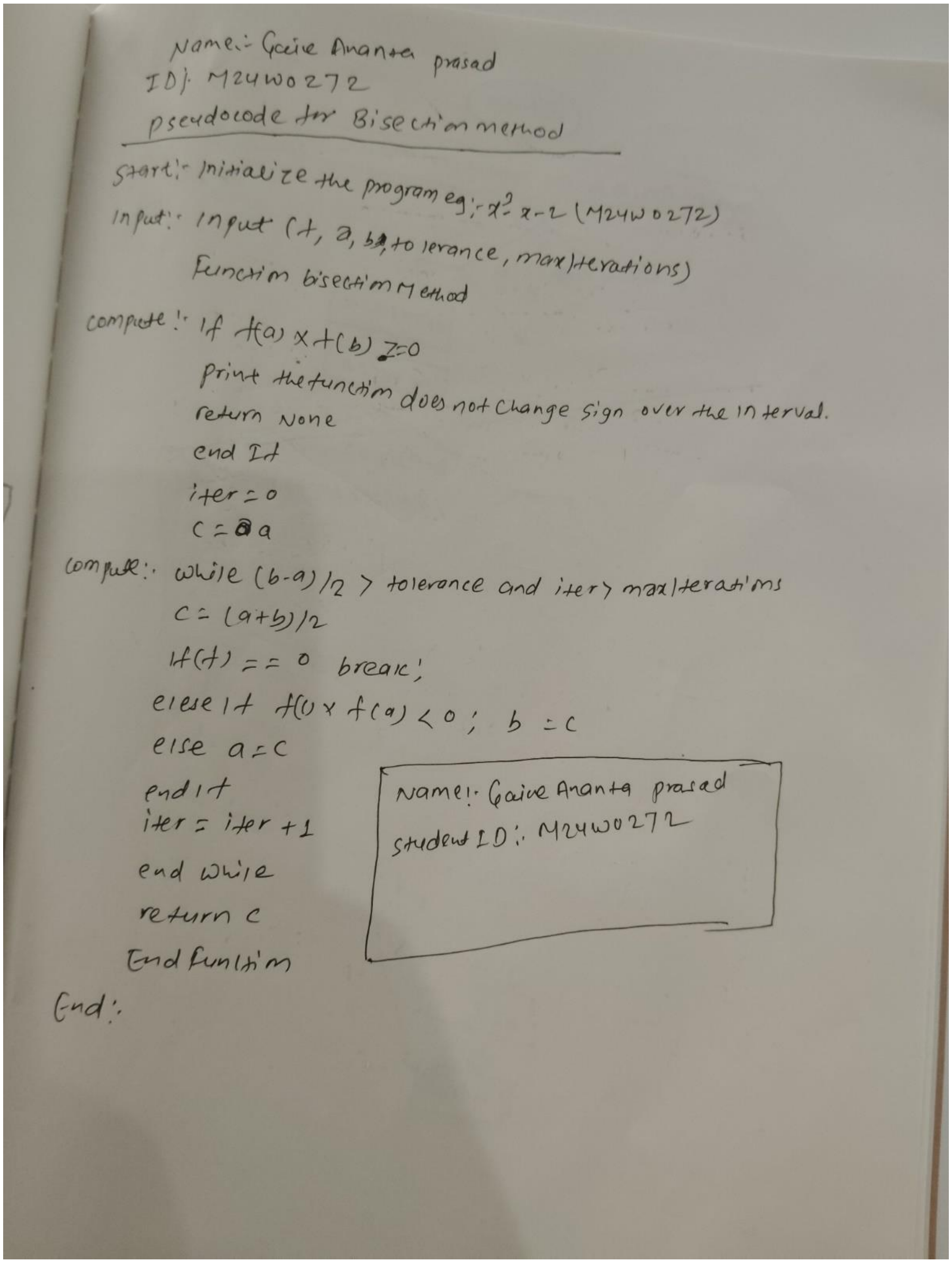
```
PS D:\kcgi\Java>  & 'C:\Users\gaire\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.10.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\gaire\AppData\Roaming\Code\User\workspaceStorage\cff8322bbc836dc0e92005c5cab87794\redhat.java\jdt_ws\Java_c3b3109e\bin' 'newtonsMethod'
Root: 1.5213798059647863
PS D:\kcgi\Java> 
```

← output

B. FLOW-CHART FOR BISECTION METHOD


* Flowchart for Bisection Method eg: x³ - x - 2
Start
input f(x): a, b, tol, maxIterations
if f(a)×f(b) ≥ 0?
yes
NO
double c:a
Name: Gaire Ananta prasad
student ID: M24W0272
int iter =0
(b-a)/2.0 > tolerance && iter <maxIteration
NO
yes
The function does not change sign over the interval
c: (a+b)/2.0
f(c)==0.0
yes
NO
LOOP
NO
f(c)×f(a)<0
yes
c
value (Root)
a=c
b=c
iter++
Print output (c) (root)
end

C. PSEUDOCODE FOR BISECTION METHOD



Name:- Gaire Ananta prasad
ID: M24W0272

pseudocode for Bisection method

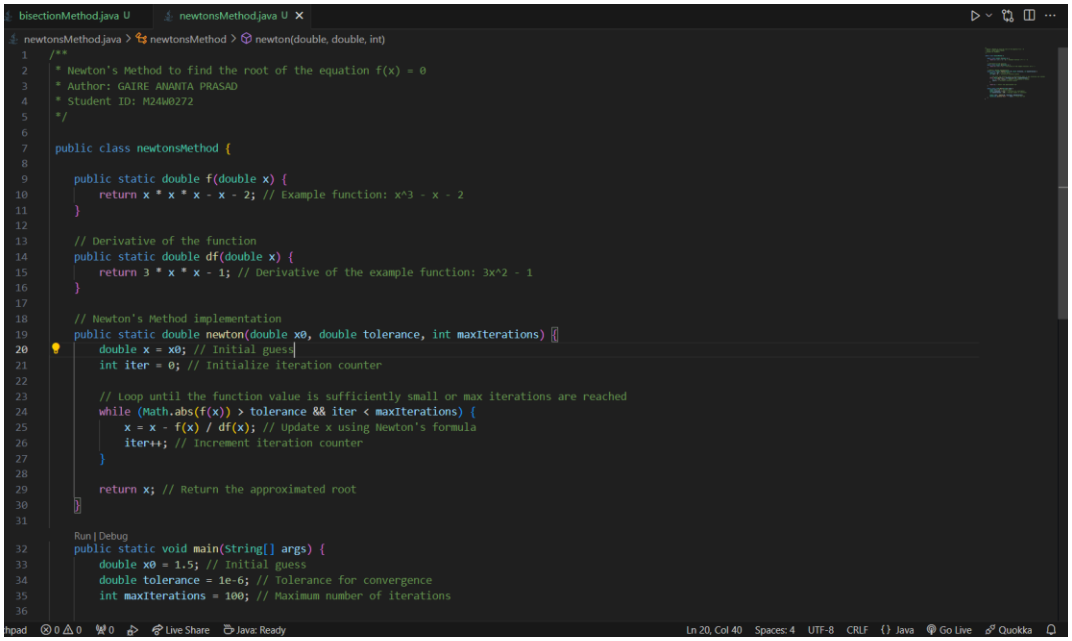
```
Start:- Initialize the program eg:- $x^2-x-2$ (M24W0272)
Input:- input (f, a, b, tolerance, maxIterations)
        Function bisectionMethod
compute:- If $f(a) \times f(b) \geq 0$
            Print the function does not change sign over the interval.
            return None
            end If
            iter = 0
            c = a
compute:- while $(b-a)/2 >$ tolerance and iter > maxIterations
            c = $(a+b)/2$
            if(f) == 0 break;
          elese if $f(c) \times f(a) < 0$; b = c
          else a = c
          endif
          iter = iter + 1
          end while
          return c
        End function
End:
```

Name:- Gaire Ananta prasad
Student ID: M24W0272

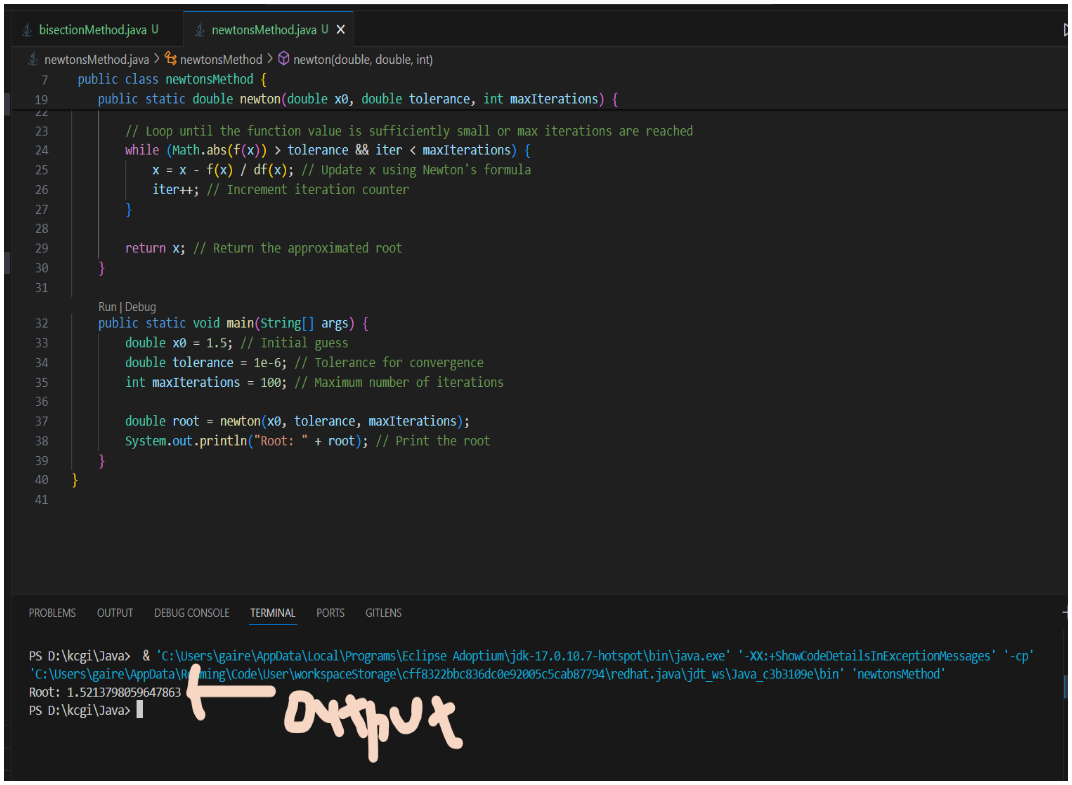
GAIRE ANANTA PRASAD
M24W0272

2. NETWON’S METHOD

A. CODE FOR NETWON’S METHOD



```java
/**
 * Newton's Method to find the root of the equation f(x) = 0
 * Author: GAIRE ANANTA PRASAD
 * Student ID: M24W0272
 */

public class newtonsMethod {

    public static double f(double x) {
        return x * x * x - x - 2; // Example function: x^3 - x - 2
    }

    // Derivative of the function
    public static double df(double x) {
        return 3 * x * x - 1; // Derivative of the example function: 3x^2 - 1
    }

    // Newton's Method implementation
    public static double newton(double x0, double tolerance, int maxIterations) {
        double x = x0; // Initial guess
        int iter = 0; // Initialize iteration counter

        // Loop until the function value is sufficiently small or max iterations are reached
        while (Math.abs(f(x)) > tolerance && iter < maxIterations) {
            x = x - f(x) / df(x); // Update x using Newton's formula
            iter++; // Increment iteration counter
        }

        return x; // Return the approximated root
    }

    public static void main(String[] args) {
        double x0 = 1.5; // Initial guess
        double tolerance = 1e-6; // Tolerance for convergence
        int maxIterations = 100; // Maximum number of iterations

```

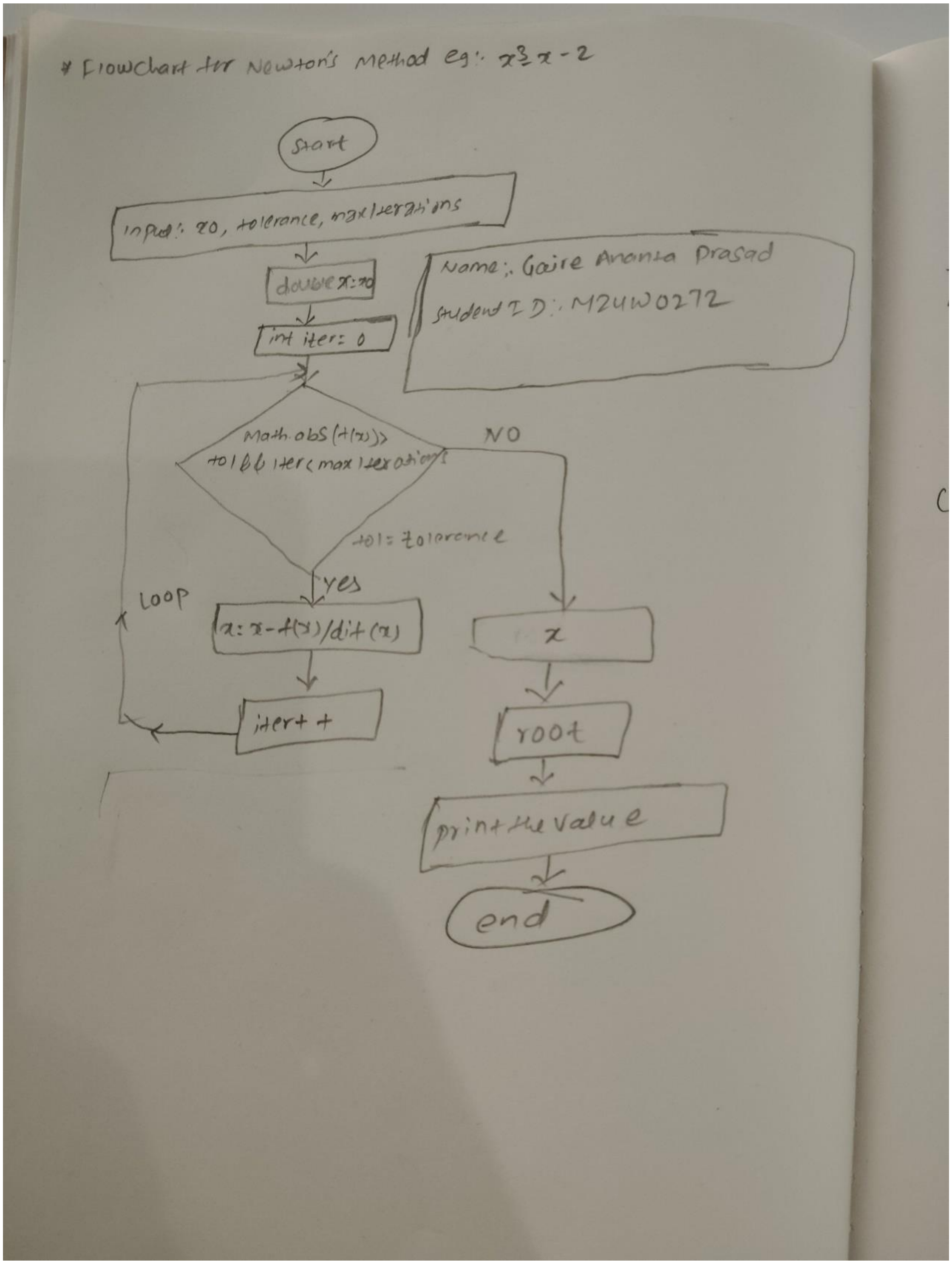


```java
public class newtonsMethod {
    public static double newton(double x0, double tolerance, int maxIterations) {
        // Loop until the function value is sufficiently small or max iterations are reached
        while (Math.abs(f(x)) > tolerance && iter < maxIterations) {
            x = x - f(x) / df(x); // Update x using Newton's formula
            iter++; // Increment iteration counter
        }

        return x; // Return the approximated root
    }

    public static void main(String[] args) {
        double x0 = 1.5; // Initial guess
        double tolerance = 1e-6; // Tolerance for convergence
        int maxIterations = 100; // Maximum number of iterations

        double root = newton(x0, tolerance, maxIterations);
        System.out.println("Root: " + root); // Print the root
    }
}

```

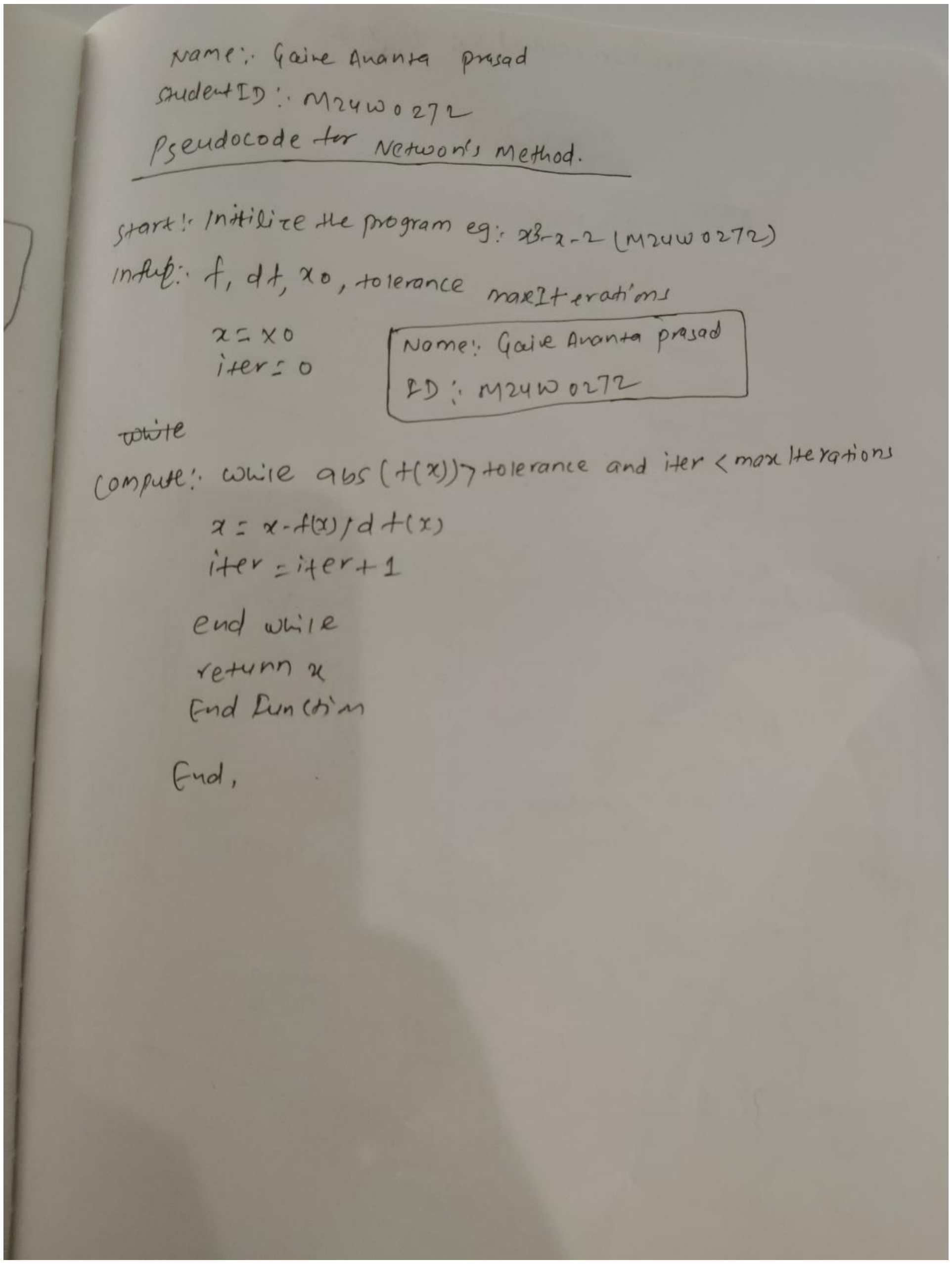
```
PS D:\kcgi\Java>  & 'C:\Users\gaire\AppData\Local\Programs\Eclipse Adoptium\jdk-17.0.10.7-hotspot\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp'
'C:\Users\gaire\AppData\Roaming\Code\User\workspaceStorage\cff8322bbc836dc0e92005c5cab87794\redhat.java\jdt_ws\Java_c3b3109e\bin' 'newtonsMethod'
Root: 1.5213798059647863
PS D:\kcgi\Java> 
```

Output

B. FLOW-CHART FOR NETWON'S METHOD


# Flowchart for Newton's Method eg: $x^3 = x - 2$
Start
input: x0, tolerance, maxIterations
double x: x0
int iter: 0
Name: Gaire Ananta Prasad
Student I D: M24W0272
Math.abs(f(x))> tol && iter< maxIterations
NO
tol= tolerance
yes
Loop
x: x-f(x)/dif(x)
iter++
x
root
print the value
end

C. PSEUDOCODE FOR NETWON'S METHOD



Name:: Gaire Ananta prasad
StudentID:: M24W0272
Pseudocode for Netwon's Method.

```
start:: Initilize the program eg:: $x^3-x-2$ (M24W0272)
input:: f, df, x0, tolerance maxIterations
        x = x0
        iter = 0
~~white~~
compute:: while abs(f(x)) > tolerance and iter < maxIterations
        x = x - f(x) / df(x)
        iter = iter + 1
      end while
      return x
      End function

     End,
```

Name:: Gaie Ananta prasad
ID:: M24W0272