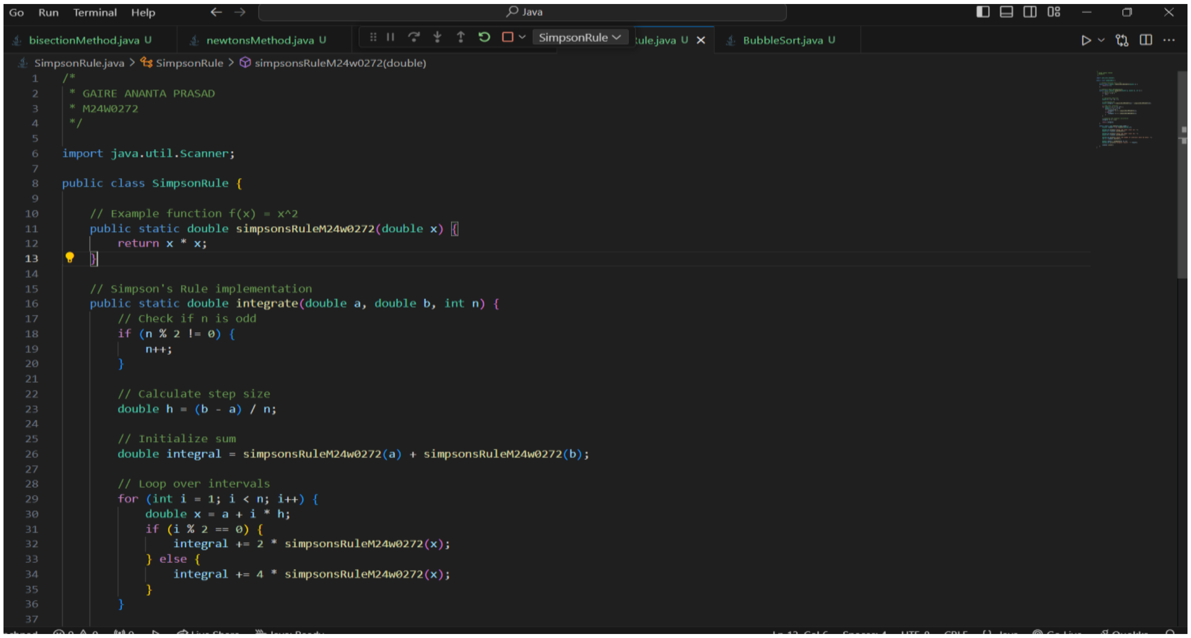
GAIRE ANANTA PRASAD

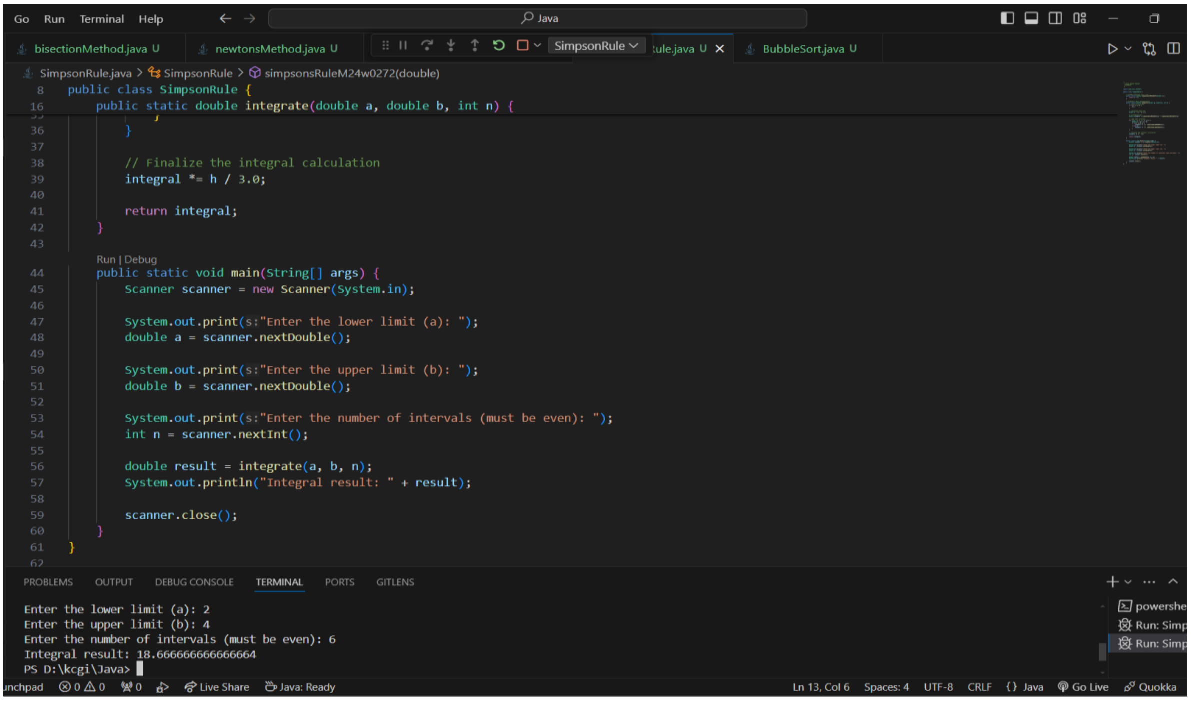
M24W0272

1. SIMPSON'S RULE

   A. Java code and output



```java
/*
 * GAIRE ANANTA PRASAD
 * M24W0272
 */

import java.util.Scanner;

public class SimpsonRule {

    // Example function f(x) = x^2
    public static double simpsonsRuleM24w0272(double x) {
        return x * x;
    }

    // Simpson's Rule implementation
    public static double integrate(double a, double b, int n) {
        // Check if n is odd
        if (n % 2 != 0) {
            n++;
        }

        // Calculate step size
        double h = (b - a) / n;

        // Initialize sum
        double integral = simpsonsRuleM24w0272(a) + simpsonsRuleM24w0272(b);

        // Loop over intervals
        for (int i = 1; i < n; i++) {
            double x = a + i * h;
            if (i % 2 == 0) {
                integral += 2 * simpsonsRuleM24w0272(x);
            } else {
                integral += 4 * simpsonsRuleM24w0272(x);
            }
        }

```


Go Run Terminal Help
Java
bisectionMethod.java U
newtonsMethod.java U
SimpsonRule
ule.java U
BubbleSort.java U
SimpsonRule.java > SimpsonRule > simpsonsRuleM24w0272(double)
Run | Debug
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
powershe
Run: Simp
Run: Simp
unchpad 0 0 0 Live Share Java: Ready
Ln 13, Col 6 Spaces: 4 UTF-8 CRLF {} Java Go Live Quokka

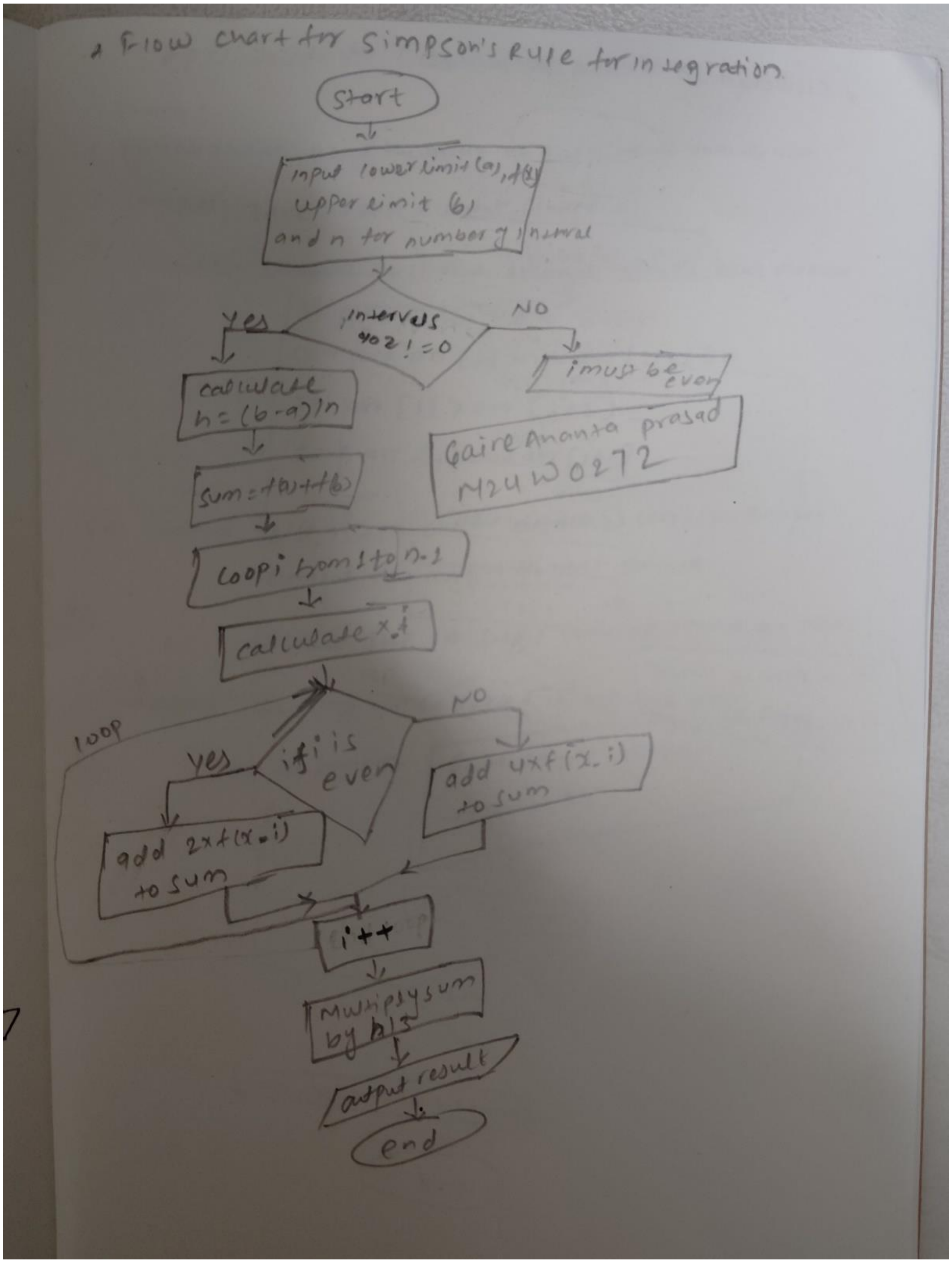
```java
public class SimpsonRule {
    public static double integrate(double a, double b, int n) {
        }

        // Finalize the integral calculation
        integral *= h / 3.0;

        return integral;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Enter the lower limit (a): ");
        double a = scanner.nextDouble();

        System.out.print(s:"Enter the upper limit (b): ");
        double b = scanner.nextDouble();

        System.out.print(s:"Enter the number of intervals (must be even): ");
        int n = scanner.nextInt();

        double result = integrate(a, b, n);
        System.out.println("Integral result: " + result);

        scanner.close();
    }
}
```

```
Enter the lower limit (a): 2
Enter the upper limit (b): 4
Enter the number of intervals (must be even): 6
Integral result: 18.666666666666664
PS D:\kcgi\Java>
```

GAIRE ANANTA PRASAD

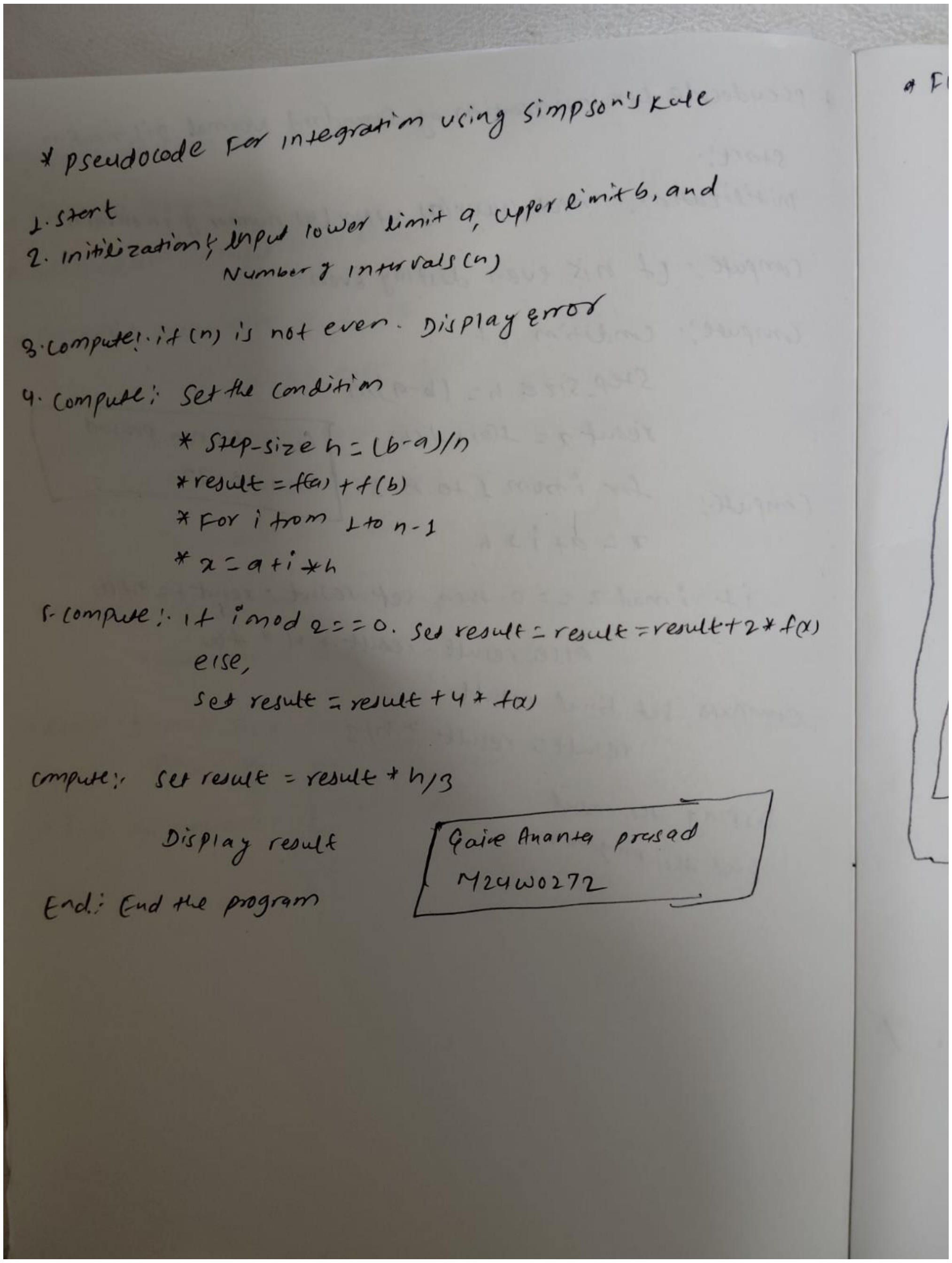
M24W0272

B. Flow-chart



Flow chart for Simpson's Rule for integration.
Start
input lower limit (a), f(x)
upper limit (b)
and n for number of interval
intervals %2 != 0
Yes
NO
i must be even
Gaire Ananta prasad
M24W0272
calculate
h=(b-a)/n
sum=f(a)+f(b)
Loop i from 1 to n-1
calculate x_i
loop
if i is even
yes
NO
add 2×f(x_i)
to sum
add 4×f(x_i)
to sum
i++
Multiply sum
by h/3
output result
end

C. Pseudocode

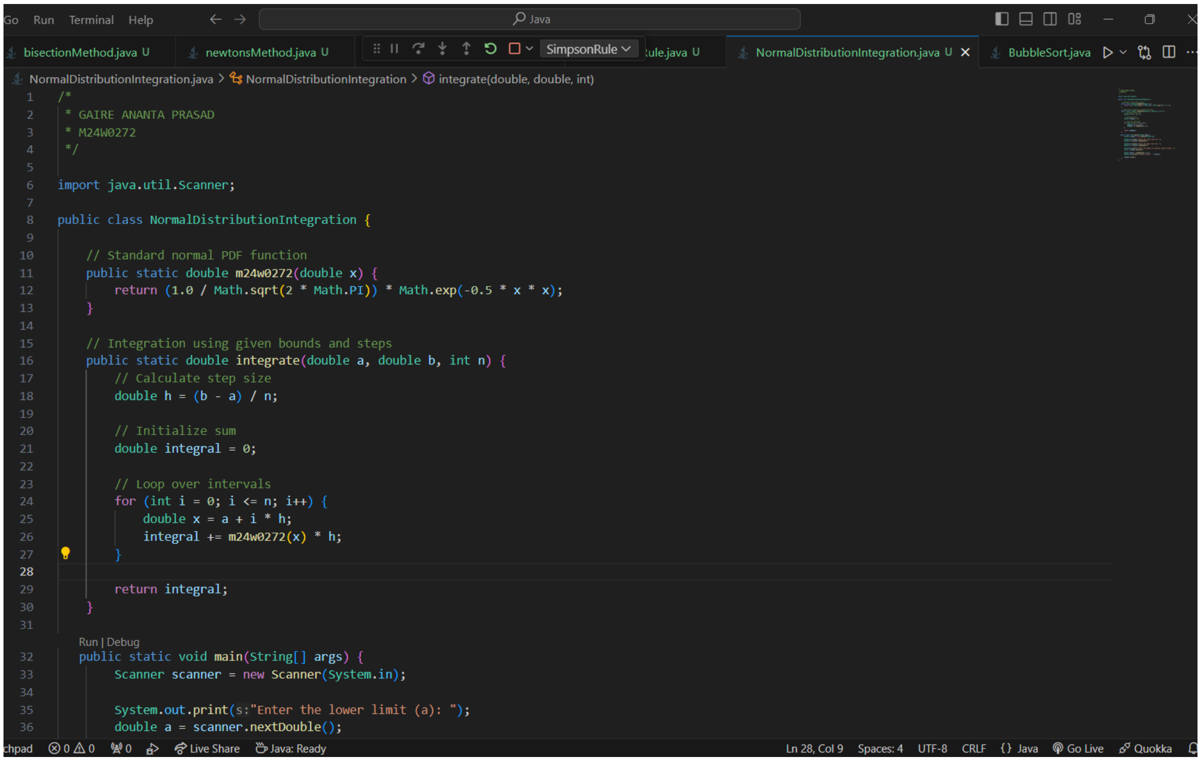


```
* pseudocode For integratim using simpson's Rule

1. start
2. initilization; input lower limit a, upper limit b, and
                  Number of intervals (n)

3. compute!. if (n) is not even. Display Error

4. compute; Set the condition
        * Step-size h = (b-a)/n
        * result = f(a) + f(b)
        * For i from 1 to n-1
        * x = a + i * h

5. compute!. if i mod 2 = = 0. Set result = result = result + 2 * f(x)
        else,
        set result = result + 4 * f(x)

compute;. set result = result * h/3

        Display result

End: End the program
```

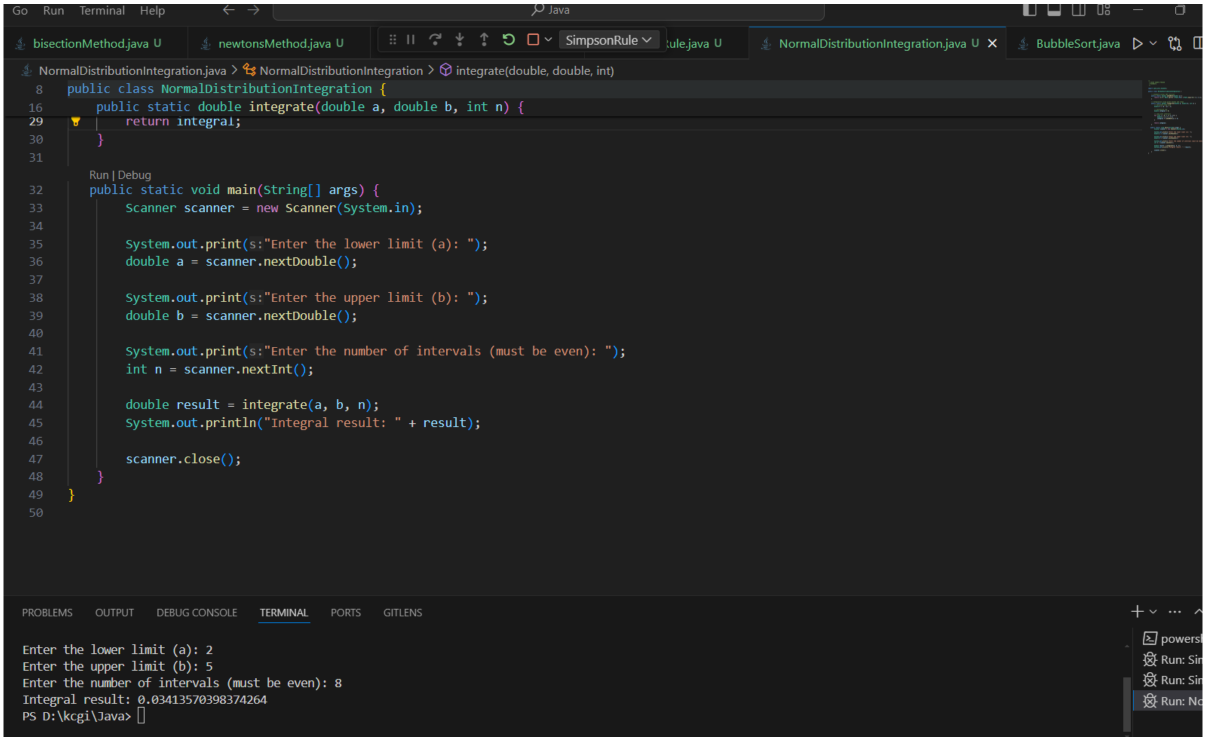
Gaire Ananta prasad
M24W0272

2. INTEGRATION OF STANDARD NORMAL DISTRIBUTION

A. Java code and Output


Go Run Terminal Help
Java
bisectionMethod.java U
newtonsMethod.java U
SimpsonRule
NormalDistributionIntegration.java U
BubbleSort.java
NormalDistributionIntegration.java > NormalDistributionIntegration > integrate(double, double, int)
Run | Debug
Ln 28, Col 9 Spaces: 4 UTF-8 CRLF {} Java Go Live Quokka
Live Share Java: Ready

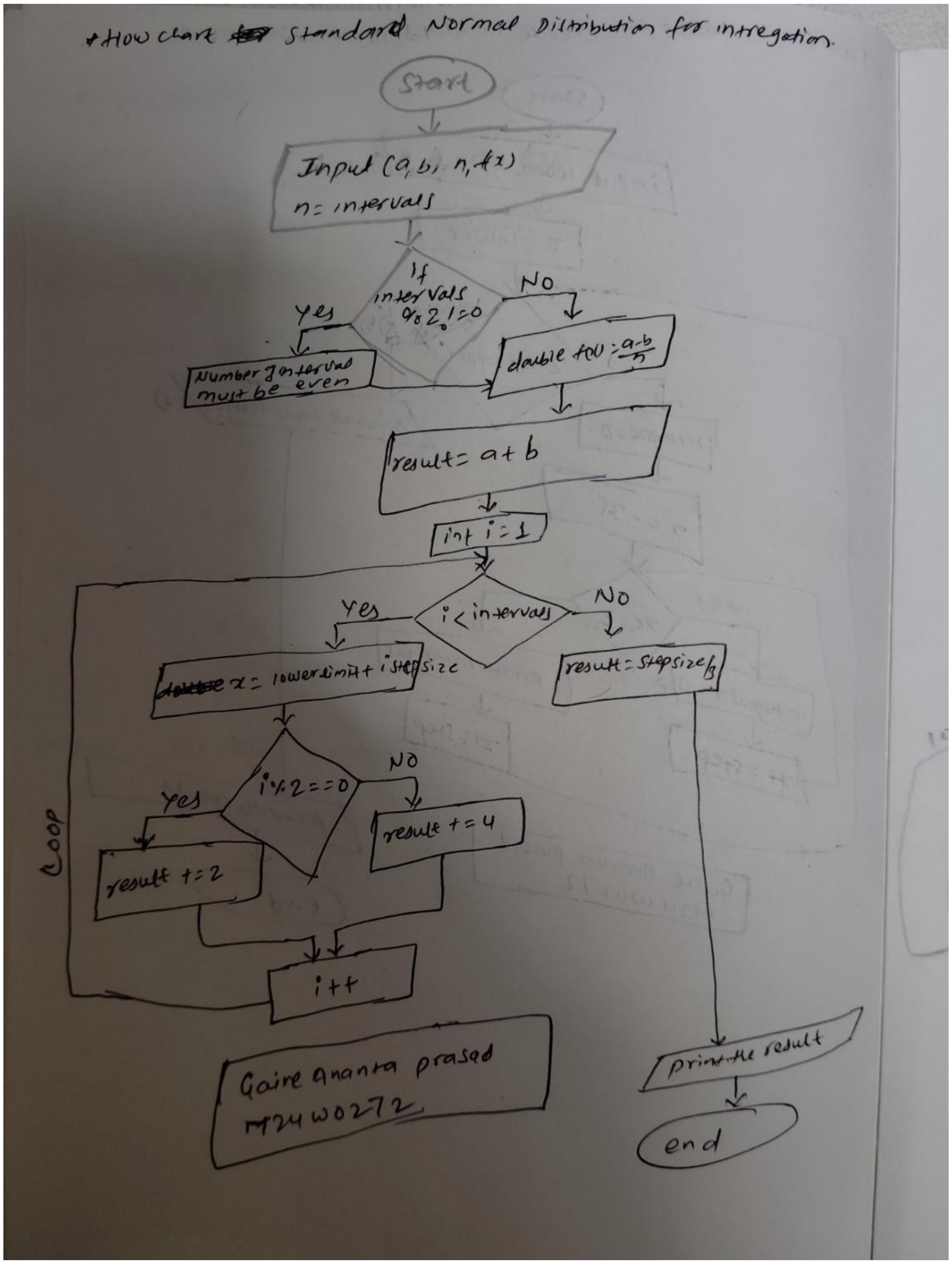
```java
/*
 * GAIRE ANANTA PRASAD
 * M24W0272
 */

import java.util.Scanner;

public class NormalDistributionIntegration {

    // Standard normal PDF function
    public static double m24w0272(double x) {
        return (1.0 / Math.sqrt(2 * Math.PI)) * Math.exp(-0.5 * x * x);
    }

    // Integration using given bounds and steps
    public static double integrate(double a, double b, int n) {
        // Calculate step size
        double h = (b - a) / n;

        // Initialize sum
        double integral = 0;

        // Loop over intervals
        for (int i = 0; i <= n; i++) {
            double x = a + i * h;
            integral += m24w0272(x) * h;
        }

        return integral;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Enter the lower limit (a): ");
        double a = scanner.nextDouble();
```


Go Run Terminal Help
Java
bisectionMethod.java U
newtonsMethod.java U
SimpsonRule
NormalDistributionIntegration.java U
BubbleSort.java
NormalDistributionIntegration.java > NormalDistributionIntegration > integrate(double, double, int)
Run | Debug
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
powers
Run: Sir
Run: Sir
Run: No

```java
public class NormalDistributionIntegration {
    public static double integrate(double a, double b, int n) {
        return integral;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Enter the lower limit (a): ");
        double a = scanner.nextDouble();

        System.out.print(s:"Enter the upper limit (b): ");
        double b = scanner.nextDouble();

        System.out.print(s:"Enter the number of intervals (must be even): ");
        int n = scanner.nextInt();

        double result = integrate(a, b, n);
        System.out.println("Integral result: " + result);

        scanner.close();
    }
}
```

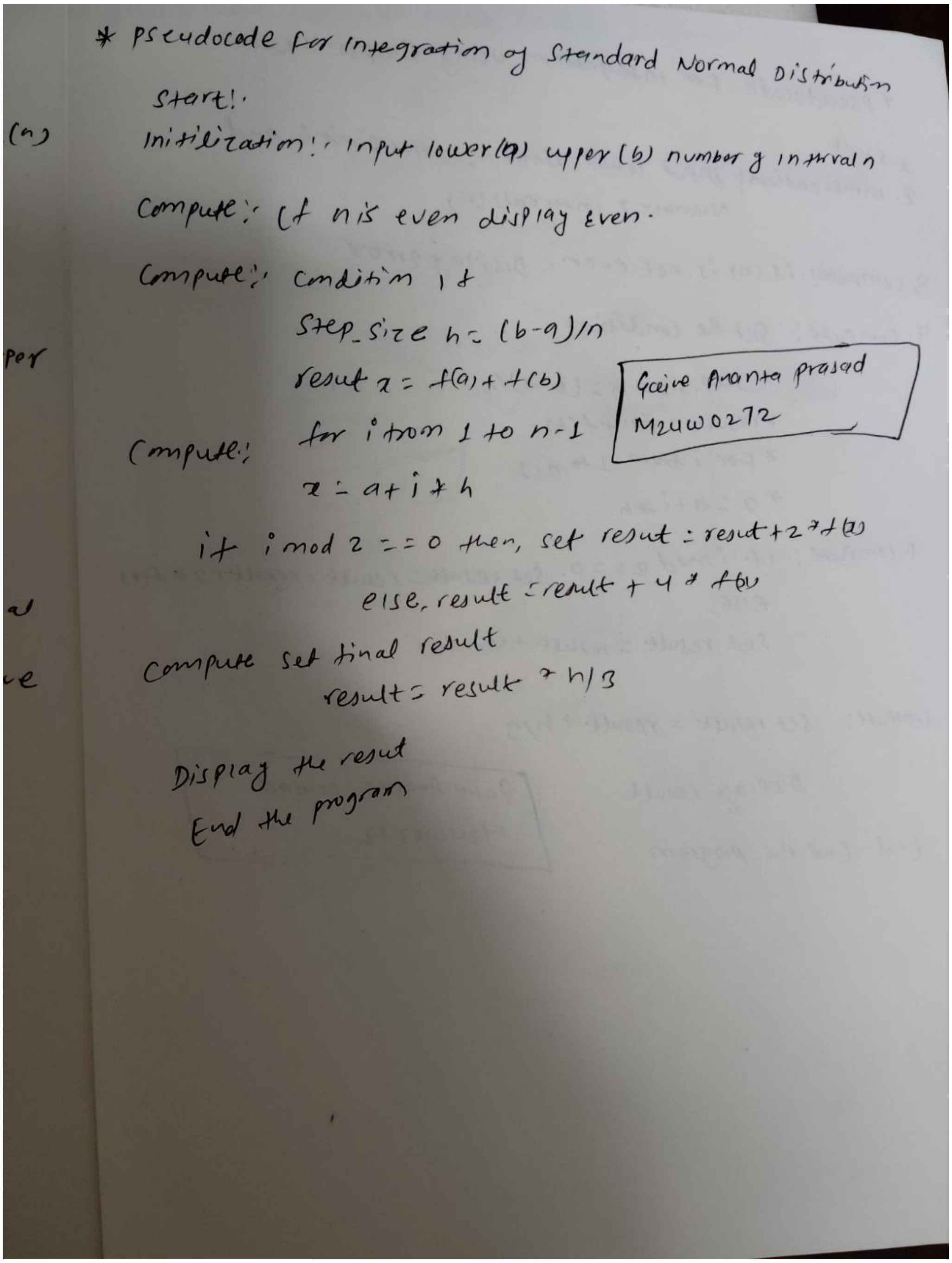
```
Enter the lower limit (a): 2
Enter the upper limit (b): 5
Enter the number of intervals (must be even): 8
Integral result: 0.03413570398374264
PS D:\kcgi\Java> 
```

B. Flow chart



GAIRE ANANTA PRASAD
M24W0272

C. Pseudocode



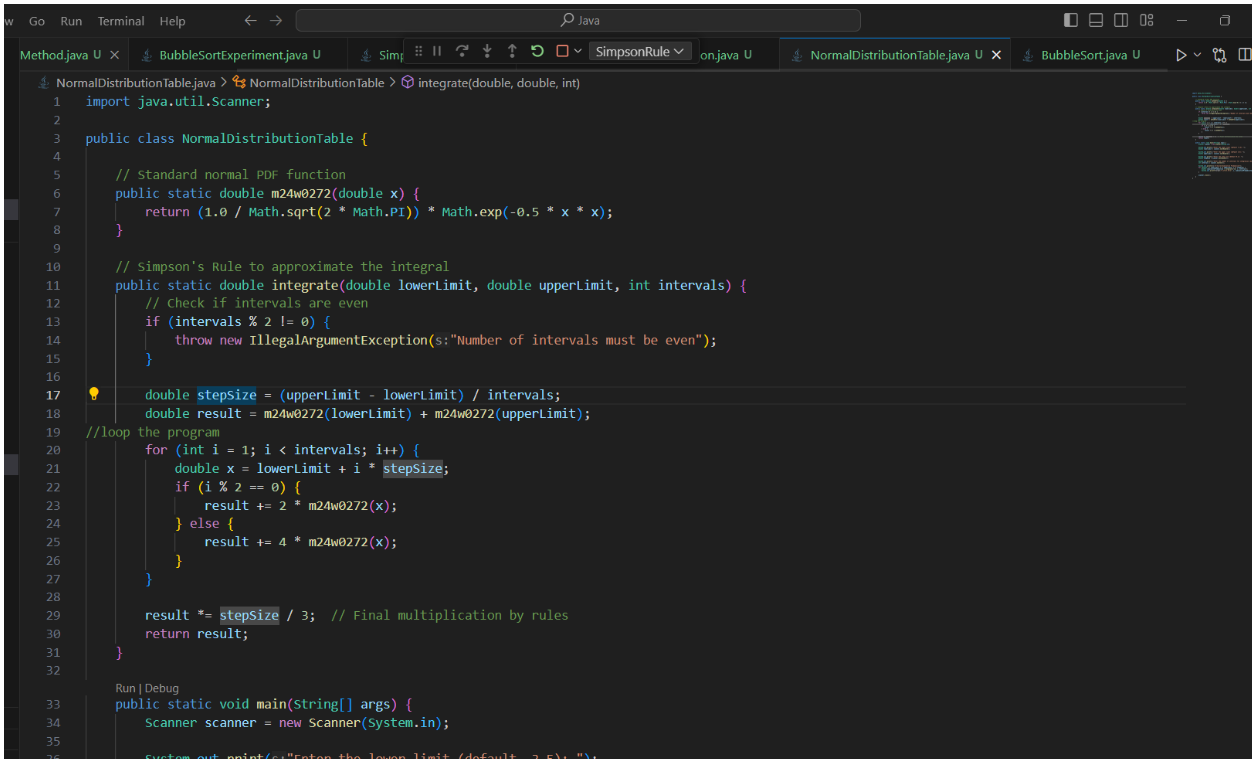
```
* Pseudocode for Integration of Standard Normal Distribution
  Start!.
  Initilization!. input lower(a) upper (b) number of interval n
  Compute:: If n is even display Even.
  Compute:: condition 1f
                Step_size $h = (b-a)/n$
                resut $x = f(a) + f(b)$
  Compute!  for i from 1 to n-1
                $x = a + i * h$
        if i mod 2 == 0 then, set resut = resut + 2 * f(x)
                  else, result = result + 4 * f(x)
  compute set final result
            result = result * h/3

   Display the resut
   End the program
```

Gaire Ananta Prasad
M24W0272

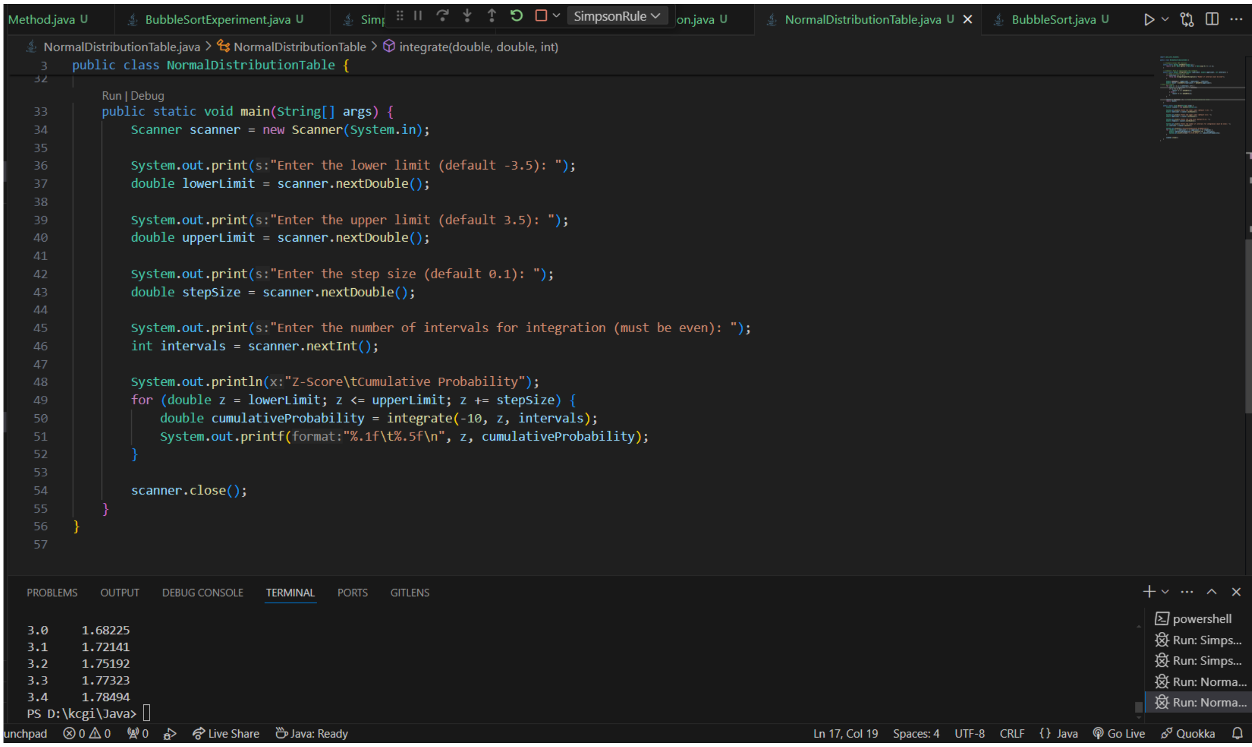
GAIRE ANANTA PRASAD
M24W0272

3. Table for standard normal distribution

   A. Java code and output


w Go Run Terminal Help
Java
Method.java U
BubbleSortExperiment.java U
Simp
SimpsonRule
on.java U
NormalDistributionTable.java U
BubbleSort.java U
NormalDistributionTable.java > NormalDistributionTable > integrate(double, double, int)
Run | Debug

```java
import java.util.Scanner;

public class NormalDistributionTable {

    // Standard normal PDF function
    public static double m24w0272(double x) {
        return (1.0 / Math.sqrt(2 * Math.PI)) * Math.exp(-0.5 * x * x);
    }

    // Simpson's Rule to approximate the integral
    public static double integrate(double lowerLimit, double upperLimit, int intervals) {
        // Check if intervals are even
        if (intervals % 2 != 0) {
            throw new IllegalArgumentException(s:"Number of intervals must be even");
        }

        double stepSize = (upperLimit - lowerLimit) / intervals;
        double result = m24w0272(lowerLimit) + m24w0272(upperLimit);
//loop the program
        for (int i = 1; i < intervals; i++) {
            double x = lowerLimit + i * stepSize;
            if (i % 2 == 0) {
                result += 2 * m24w0272(x);
            } else {
                result += 4 * m24w0272(x);
            }
        }

        result *= stepSize / 3;  // Final multiplication by rules
        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

```

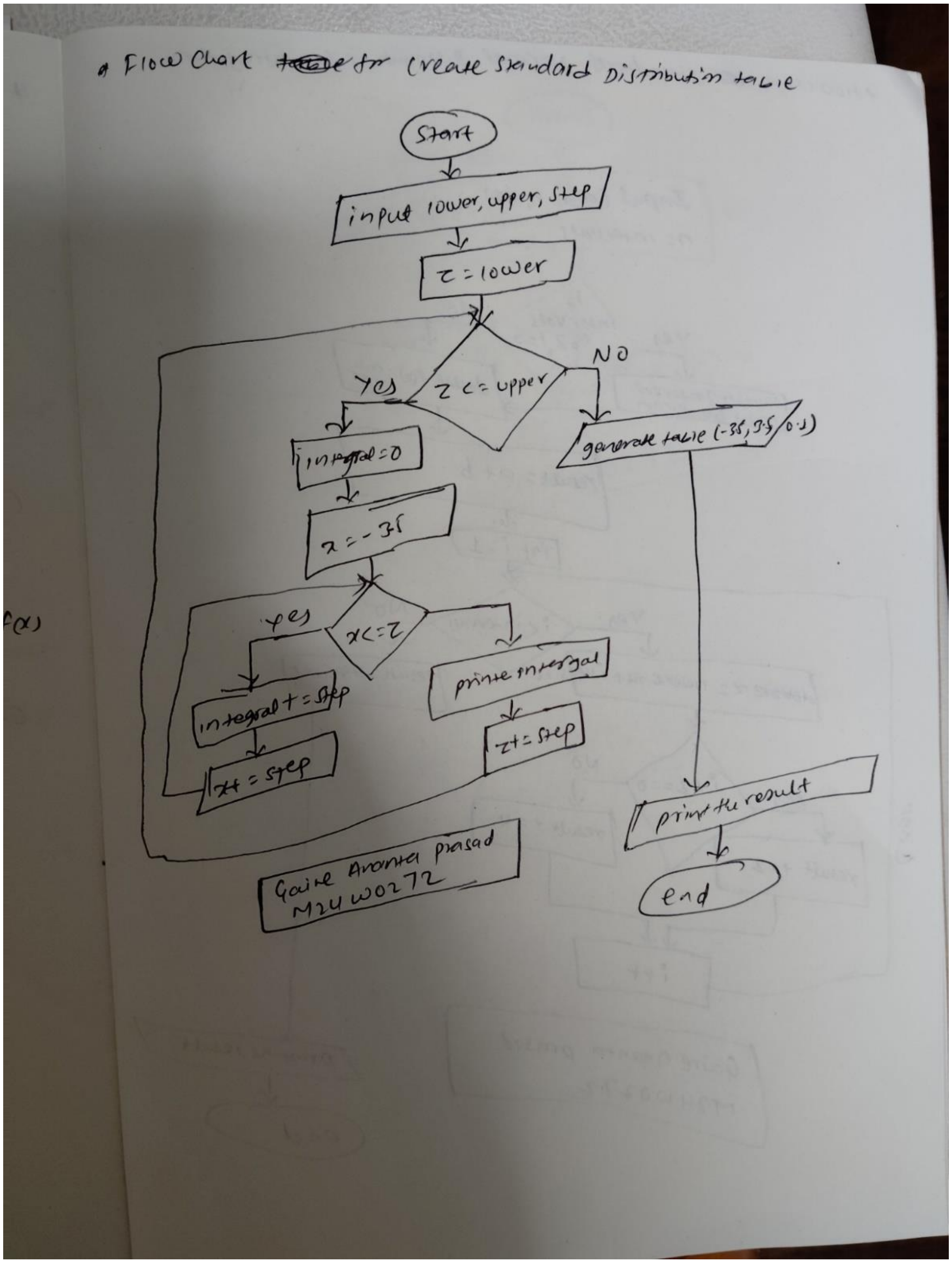


```java
public class NormalDistributionTable {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print(s:"Enter the lower limit (default -3.5): ");
        double lowerLimit = scanner.nextDouble();

        System.out.print(s:"Enter the upper limit (default 3.5): ");
        double upperLimit = scanner.nextDouble();

        System.out.print(s:"Enter the step size (default 0.1): ");
        double stepSize = scanner.nextDouble();

        System.out.print(s:"Enter the number of intervals for integration (must be even): ");
        int intervals = scanner.nextInt();

        System.out.println(x:"Z-Score\tCumulative Probability");
        for (double z = lowerLimit; z <= upperLimit; z += stepSize) {
            double cumulativeProbability = integrate(-10, z, intervals);
            System.out.printf(format:"%.1f\t%.5f\n", z, cumulativeProbability);
        }

        scanner.close();
    }
}

```

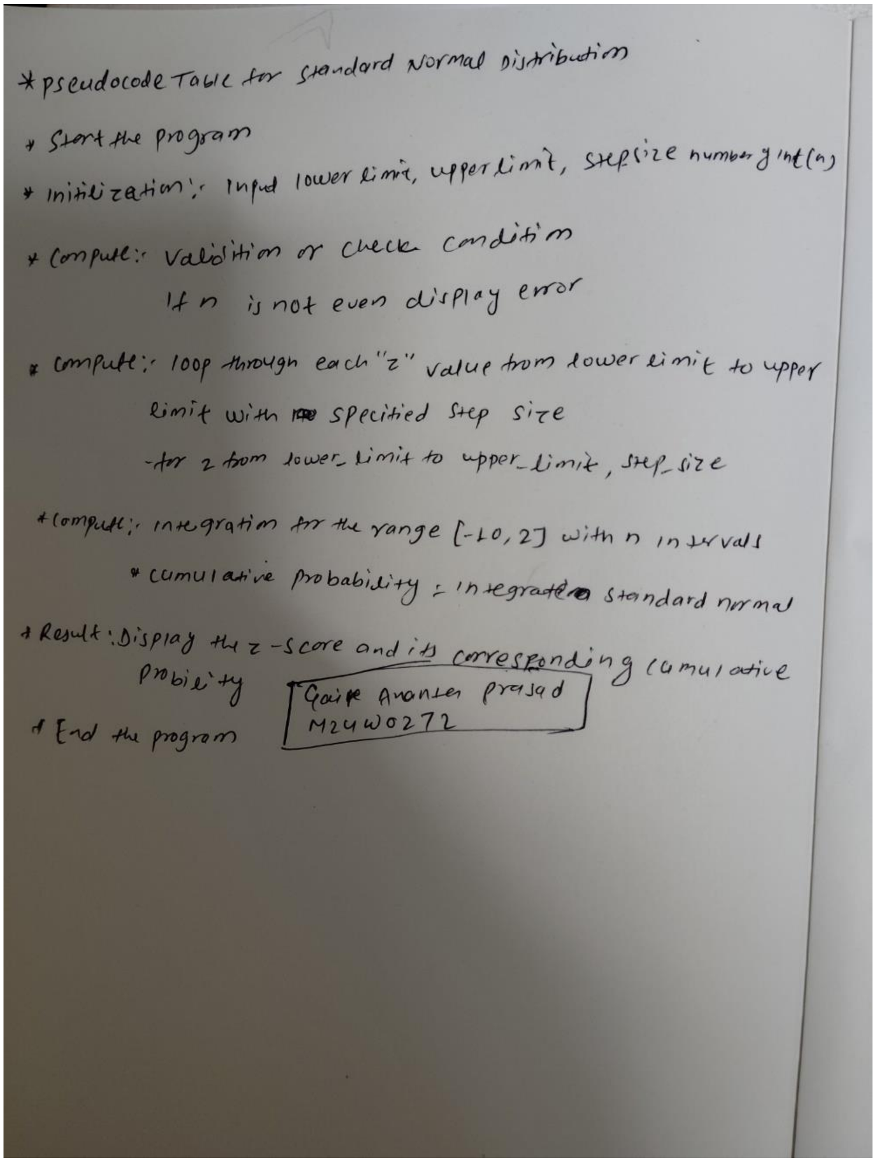
```
3.0     1.68225
3.1     1.72141
3.2     1.75192
3.3     1.77323
3.4     1.78494
PS D:\kcgi\Java> 
```

GAIRE ANANTA PRASAD
M24W0272

B. Flowchart


* Flow Chart ~~table~~ for Create Standard Distribution table
Start
input lower, upper, step
z = lower
z <= upper
Yes
NO
generate table (-3.5, 3.5, 0.1)
integral=0
x = -3.5
x<=z
yes
integral+=step
x+=step
print integral
z+=step
print the result
end
Gaire Ananta Prasad
M24W0272
f(x)

C. Pseudocode



```
* pseudocode Table for Standard Normal Distribution

* Start the program
* Initilization:- Input lower limit, upper limit, stepsize number of int(n)

* Compute:- Validition or Check condition
            If n is not even display error

* Compute:- loop through each "z" value from lower limit to upper
            limit with specified Step size
            -for z from lower_limit to upper_limit, step_size

* Compute:- integration for the range [-10, 2] with n intervals
          * cumulative probability = integrated standard normal
* Result: Display the z-score and its corresponding cumulative
            Probility
* End the program
```

Gaire Ananta prasad
M24W0272