NAME: ABENA DUROWAA DWOBENG
INDEX NUMBER: 8547721
JAVA ASSIGNMENT

ALGORITHM

Initialize Variables and Arrays:

Create arrays to store index numbers, midterm scores, and final scores for students.
Initialize variables for total score, maximum score, and minimum score.
Create an array to store grade frequencies.
Accept Input from User:

Prompt the user to enter the number of students.
Use a loop to input data for each student, including index number, midterm score, and final score.
Validate that the midterm score is not greater than 30 and the final score is not greater than 70.
Calculate and Display Results:

Loop through each student:
Calculate the final score as the sum of the midterm and final scores.
Update total score, maximum score, and minimum score.
Determine the grade based on the KNUST grading system.
Display individual student information (index number, final score, grade).
Calculate Average, Maximum, and Minimum:

Calculate the average score by dividing the total score by the number of students.
Display the average score, maximum score, and minimum score.
Display Grade Frequencies:

Display the frequencies of each grade (A, B, C, D, F) based on the KNUST grading system.

CODE IN JAVA (PROCEDURAL):

```
import java.util.Scanner;

public class GradeCalculator {

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

        // Accept input for the number of students
        System.out.print("Enter the number of students: ");
        int numStudents = scanner.nextInt();

        // Arrays to store student data
        String[] indexNumbers = new String[numStudents];
        int[] midtermScores = new int[numStudents];
        int[] finalScores = new int[numStudents];

        // Accept input for each student with validation
        for (int i = 0; i < numStudents; i++) {
            System.out.println("\nEnter details for Student " + (i + 1));
            System.out.print("Index Number: ");
            indexNumbers[i] = scanner.next();

            // Validate and accept midterm score
```

```
            do {
                System.out.print("Midterm Score (<= 30): ");
                midtermScores[i] = scanner.nextInt();
            } while (midtermScores[i] > 30);

            // Validate and accept final exam score
            do {
                System.out.print("Final Score (<= 70): ");
                finalScores[i] = scanner.nextInt();
            } while (finalScores[i] > 70);
        }

        scanner.close();

        double totalScore = 0;
        int maxScore = Integer.MIN_VALUE;
        int minScore = Integer.MAX_VALUE;

        // Output header
        System.out.println("\nIndex No.\tFinal Score\tGrade");

        // Arrays to store grade frequencies
        int[] gradeFrequencies = new int[5]; // A, B, C, D, F

        // Loop through each student
        for (int i = 0; i < numStudents; i++) {
            // Calculate final score
            int finalScore = midtermScores[i] + finalScores[i];
            totalScore += finalScore;

            // Update max and min scores
            maxScore = Math.max(maxScore, finalScore);
            minScore = Math.min(minScore, finalScore);

            // Determine grade based on the KNUST grading system
            char grade = determineGrade(finalScore);

            // Output individual student information
            System.out.println(indexNumbers[i] + "\t\t" + finalScore + "\t\t" + grade);

            // Update grade frequencies
            updateGradeFrequencies(gradeFrequencies, grade);
        }

        // Calculate average score
        double averageScore = totalScore / numStudents;

        // Output additional information
        System.out.println("\nAverage Score: " + averageScore);
        System.out.println("Maximum Score: " + maxScore);
        System.out.println("Minimum Score: " + minScore);

        // Display grade frequencies
        displayGradeFrequencies(gradeFrequencies);
    }

    // Function to determine the grade based on the KNUST grading system
    private static char determineGrade(int finalScore) {
```

```
        if (finalScore >= 70) {
            return 'A';
        } else if (finalScore >= 60) {
            return 'B';
        } else if (finalScore >= 50) {
            return 'C';
        } else if (finalScore >= 40) {
            return 'D';
        } else {
            return 'F';
        }
    }

    // Function to update grade frequencies
    private static void updateGradeFrequencies(int[] gradeFrequencies, char grade) {
        switch (grade) {
            case 'A':
                gradeFrequencies[0]++;
                break;
            case 'B':
                gradeFrequencies[1]++;
                break;
            case 'C':
                gradeFrequencies[2]++;
                break;
            case 'D':
                gradeFrequencies[3]++;
                break;
            case 'F':
                gradeFrequencies[4]++;
                break;
        }
    }

    // Function to display grade frequencies
    private static void displayGradeFrequencies(int[] gradeFrequencies) {
        System.out.println("\nGrade Frequencies:");
        System.out.println("A: " + gradeFrequencies[0]);
        System.out.println("B: " + gradeFrequencies[1]);
        System.out.println("C: " + gradeFrequencies[2]);
        System.out.println("D: " + gradeFrequencies[3]);
        System.out.println("F: " + gradeFrequencies[4]);
    }
}
```