CS2043

Assignment 2

October 5th, 2023

UNB Fredericton

Class Code: CS2043

Document: Assignment 2

Student Name: Will Ross

Student Number: #3734692

Due Date: October 9th, 2023

Exercise 1

a) Provide two references (web, library, textbooks) defining rational numbers and operations on rational numbers.

two links about rational numbers
https://www.mathsisfun.com/rational-numbers.html
https://www.freecodecamp.org/news/what-is-a-rational-number-definition-and-rational-number-example/

b) Set the requirements for the inputs to your program.

set the requirements as an int for my scanner class.

c) Set the requirements for the outputs from your program (you may consider using "irreducible fraction" representation when specifying this requirement).

set the requirements for the output as an int.

d) Set the requirements for the representation of fractional numbers in the program.

The output when in fraction form will be shown by having the numerator and denominator separated till the end.

e) Set the requirements for how the arithmetic operations need to be performed.

The main operation needed to happen is the average arithmetic, which is:

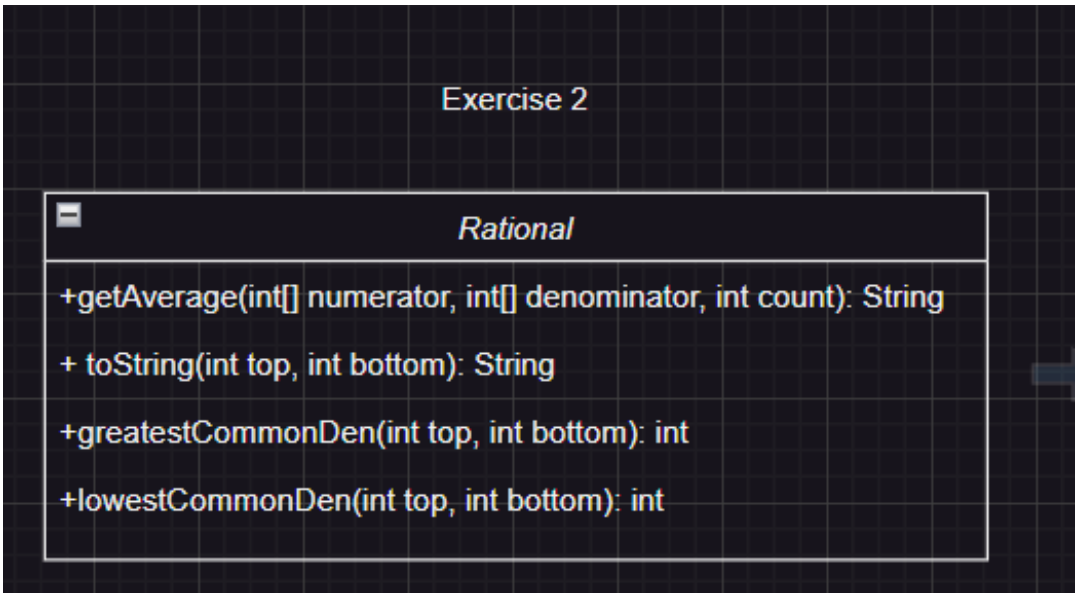
$$\frac{\text{Numerator} * \text{denominator lowest common multiple}}{(\text{Lowest common multiple} * \text{Denominator}) * \text{amount of inputs}}$$

Exercise 2

a) Estimate how much effort it would take to complete this exercise. State the composition of your estimate (e.g. requirements 1 hour (from Exercise 1 above), design 1 hour, coding 2 hours, testing 1 hour, documentation 1 hour for the total of 6 hour).

Type of activity	Time
Understanding the problem	30 mins
Identifying the structure	25 mins
Coding	1 hour 30 mins
Testing (data set, compiling, error issues)	25 mins
Documentation	25 mins
Total time of project	**3 hours and 15 minutes**

b) Show the design of your solution in terms of class diagrams as in Lab 1.



c) Code and test your program. Include the source code and the results of testing in your assignment report

```
import java.util.Scanner;

public class Rational{
   public static void main(String[] agrs) {
      int numerator[];
      int denominator[];

      Scanner sc = new Scanner(System.in);
      System.out.println("How many inputs would you like to enter?");
      int length = sc.nextInt();
      sc.nextLine();

      numerator = new int[length];
      denominator = new int[length];
      System.out.println("Enter rational number in the format (numerator/denominator):\nThen press
enter to type a new number.");
      for (int i = 0; i < length; i++) {

         String current = sc.nextLine();
         String[] currentNums = current.split("/");

         if (currentNums.length != 2) {
            System.out.println("Invalid input format. Please use the format (numerator/denominator).");
            i--;
            continue;
         }
         try {
            numerator[i] = Integer.parseInt(currentNums[0]);
            denominator[i] = Integer.parseInt(currentNums[1]);
            System.out.println("Read: " + numerator[i] + "/" + denominator[i]);
         } catch (NumberFormatException e) {
            System.out.println("Invalid numeric input. Please enter valid integers for numerator and
denominator.");
            i--;
         }
      }
      String average = getAverage(numerator, denominator, length);
      System.out.println("The average rational number is: " + average);
      sc.close();
   }

   public static String toString(int top, int bottom) {
      return top + "/" + bottom;
   }
```

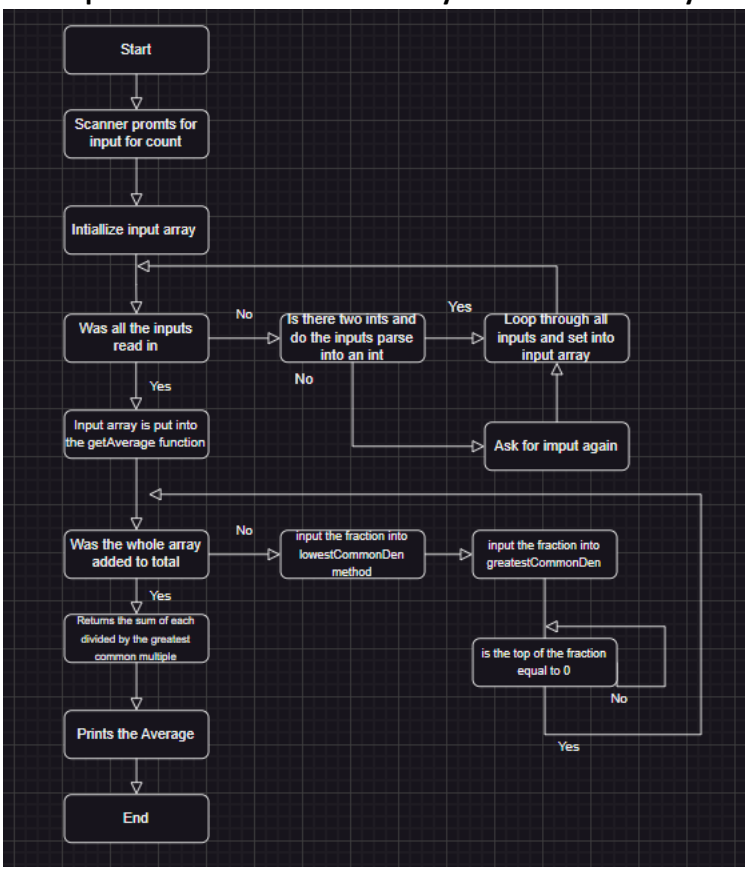
```
  private static int greatestCommonDen(int top, int bottom) {
    if (top == 0)
      return bottom;
    return greatestCommonDen(bottom % top, top);
  }

  private static int lowestCommonDen(int top, int bottom) {
    return (top * bottom) / greatestCommonDen(top, bottom);
  }

  private static String getAverage(int[] numerator, int[] denominator, int count) {
    int commonDenominator = denominator[0] * count;
    for (int i = 1; i < denominator.length; i++) {
      commonDenominator = lowestCommonDen(commonDenominator, denominator[i] * count);
    }

    int totalSumNumerator = 0;
    for (int i = 0; i < numerator.length; i++) {
      totalSumNumerator += (numerator[i] * (commonDenominator / (denominator[i] * count)));
    }

    int greatestCommonDen = greatestCommonDen(totalSumNumerator, commonDenominator);
    return toString(totalSumNumerator / greatestCommonDen, commonDenominator /
greatestCommonDen);
  }

  }
}
```

d) Compare the actual time spent to your estimates form Part (a) above.

Type of activity	Estimated Time	Actual Time

Understanding the problem	30 mins	45 mins
Identifying the structure	25 mins	25 mins
Coding	1 hour 30 mins	1 hour 45 mins
Testing (data set, compiling, error issues)	25 mins	25 mins
Documentation	25 mins	25 mins
Total time of project	**3 hours and 15 minutes**	**3 hour and 45 minutes**

e) Code analysis part: Calculate McCabe's Cyclomatic Complexity for our actual entire program developed for this Exercise 2. Show your calculations in your assignment report.



Cyclomatic complexity = E – N + 2p,
where E = number of edges of the graph,
N = number of nodes of the graph, and
p = number of connected components (usually p = 1)

E = 18
N = 15
P = 1

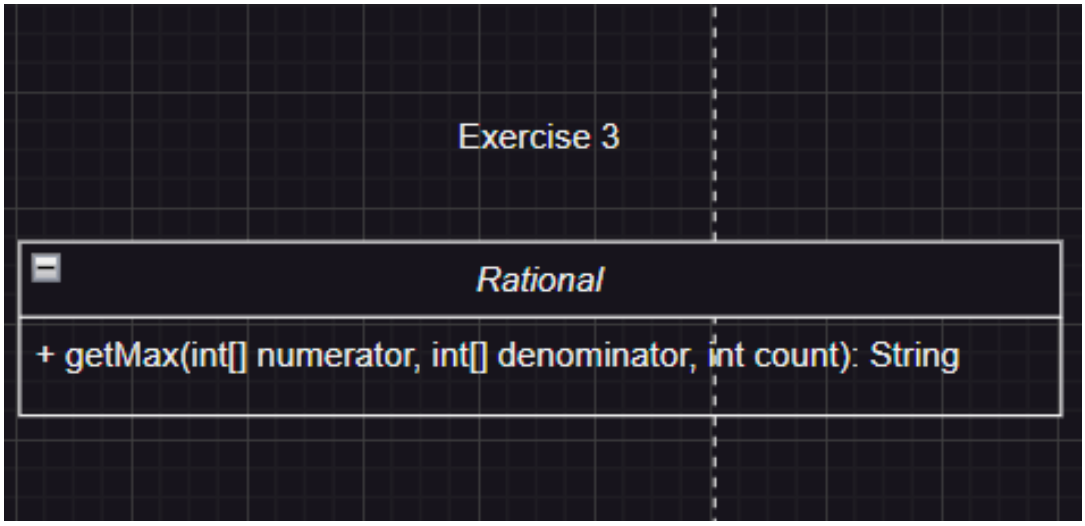
Cyclomatic complexity = 5

Exercise 3

a) Estimate how much effort it would take to complete this exercise. State the composition of your estimate.

Type of activity	Time
Understanding the problem	15 mins
Identifying the structure	15 mins
Coding	30 mins
Testing (data set, compiling, error issues)	15 mins
Documentation	25 mins
Total time of project	**1 hour and 50 minutes**

b) Show the design of your solution in terms of class diagrams as in Lab 1.



c) Code and test your program. While coding, do not modify the existing structures (loops, methods etc.) and only add new constructs (for example, do not add new statements to the loop used for calculating the average in Exercise 1). Include only the new source code added to your solution, and also the new test run results for the entire solution in your assignment report.

```
import java.util.Scanner;

public class Rational{
   public static void main(String[] agrs) {
      int numerator[];
      int denominator[];

      Scanner sc = new Scanner(System.in);
      System.out.println("How many inputs would you like to enter?");
```

```
        int length = sc.nextInt();
        sc.nextLine();

        numerator = new int[length];
        denominator = new int[length];
        System.out.println("Enter rational number in the format (numerator/denominator):\nThen press
enter to type top new number.");
        for (int i = 0; i < length; i++) {

            String current = sc.nextLine();
            String[] currentNums = current.split("/");

            if (currentNums.length != 2) {
                System.out.println("Invalid input format. Please use the format (numerator/denominator).");
                i--;
                continue;
            }
            try {
                numerator[i] = Integer.parseInt(currentNums[0]);
                denominator[i] = Integer.parseInt(currentNums[1]);
                System.out.println("Read: " + numerator[i] + "/" + denominator[i]);
            } catch (NumberFormatException e) {
                System.out.println("Invalid numeric input. Please enter valid integers for numerator and
denominator.");
                i--;
            }
        }

        String average = getAverage(numerator, denominator, length);
        System.out.println("The average rational number is: " + average);

        String max = getMax(numerator, denominator, length);
        System.out.println("The max rational number is: " + max);



        sc.close();
    }

    public static String toString(int top, int bottom) {
        return top + "/" + bottom;
    }

    private static int greatestCommonDen(int top, int bottom) {
```

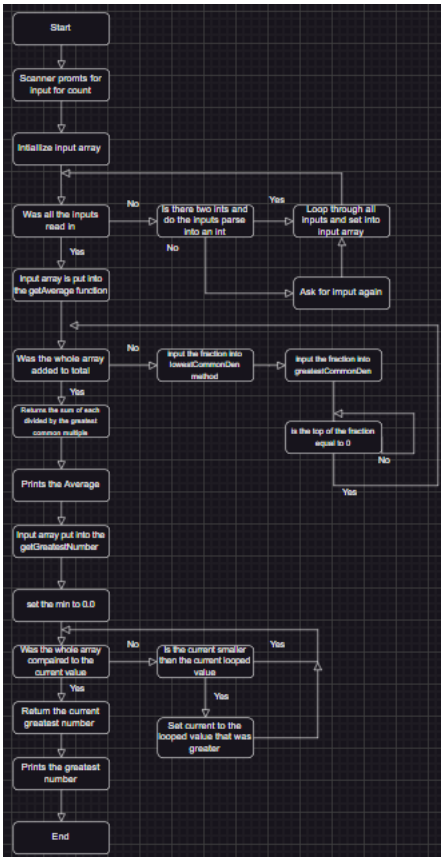
```
    if (top == 0)
      return bottom;
    return greatestCommonDen(bottom % top, top);
  }

  private static int lowestCommonDen(int top, int bottom) {
    return (top * bottom) / greatestCommonDen(top, bottom);
  }

  private static String getAverage(int[] numerator, int[] denominator, int count) {
    int commonDenominator = denominator[0] * count;
    for (int i = 1; i < denominator.length; i++) {
      commonDenominator = lowestCommonDen(commonDenominator, denominator[i] * count);
    }

    int totalSumNumerator = 0;
    for (int i = 0; i < numerator.length; i++) {
      totalSumNumerator += (numerator[i] * (commonDenominator / (denominator[i] * count)));
    }

    int greatestCommonDen = greatestCommonDen(totalSumNumerator, commonDenominator);
    return toString(totalSumNumerator / greatestCommonDen, commonDenominator /
greatestCommonDen);
  }
  private static String getMax(int[] numerator, int[] denominator, int count){
    double max = 0.0;
    int maxIndex = -1;
                for (int i = 0; i < numerator.length; i++) {
      double current = (double) numerator[i] / denominator[i];

      if (current > max) {
        max = current;
        maxIndex = i;
      }
    }
    return toString(numerator[maxIndex], denominator[maxIndex]);
  }
  }
}
```

d) Compare the actual time spent to your estimates form Part (a) above.

Type of activity	Estimated Time	Actual Time
Understanding the problem	15 mins	15 mins
Identifying the structure	15 mins	10 mins

Coding	30 mins	25 mins
Testing (data set, compiling, error issues)	15 mins	15 mins
Documentation	25 mins	20 mins
Total time of project	**1 hour and 50 minutes**	**1 hour and 35 minutes**

e) Code analysis part: Calculate McCabe's Cyclomatic Complexity for our actual entire program for this Exercise 3. Show your calculations in your assignment report.



Cyclomatic complexity = E – N + 2p,
where E = number of edges of the graph,
N = number of nodes of the graph, and
p = number of connected components (usually p = 1)
E = 26
N = 22
P = 1

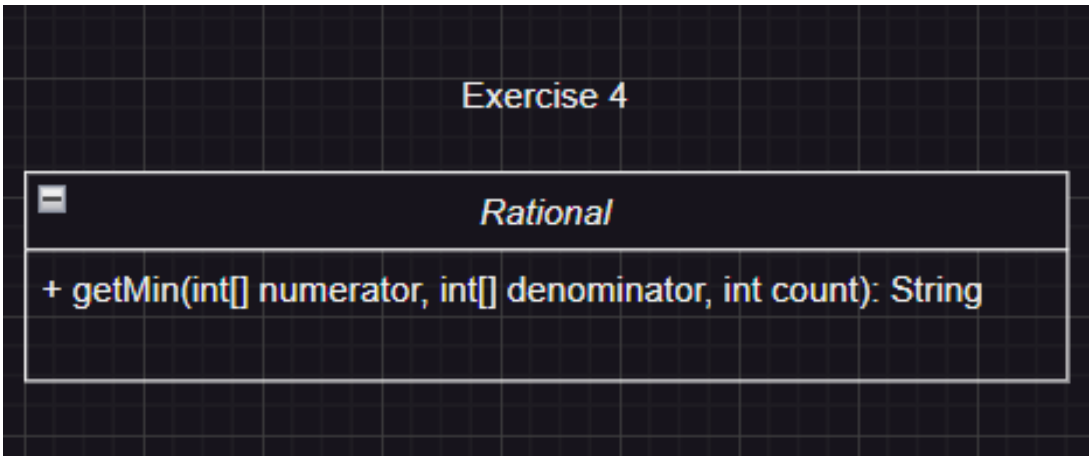
Cyclomatic complexity = 26 – 22 + 2(1)
Cyclomatic complexity = 6

Exercise 4

a) Estimate how much effort it would take to complete this exercise. State the composition of your estimate.

Type of activity	Time
Understanding the problem	10 mins
Identifying the structure	5 mins
Coding	25 mins
Testing (data set, compiling, error issues)	10 mins
Documentation	25 mins
Total time of project	**1 hour and 15 minutes**

b) Show the design of your solution in terms of class diagrams as in Lab 1.



c) Code and test your program. While coding, do not modify the existing structures (loops, methods etc.) and only add new constructs (for example, do not add new statements to the loops used in Exercise 2). Include only the new source code added to your solution, and also the new test run results for the entire solution in your assignment report.

```
import java.util.Scanner;

public class Rational{
   public static void main(String[] agrs) {
      int numerator[];
      int denominator[];

      Scanner sc = new Scanner(System.in);
      System.out.println("How many inputs would you like to enter?");
      int length = sc.nextInt();
      sc.nextLine();

      numerator = new int[length];
      denominator = new int[length];
      System.out.println("Enter rational number in the format (numerator/denominator):\nThen press
enter to type top new number.");
```

```
        for (int i = 0; i < length; i++) {

            String current = sc.nextLine();
            String[] currentNums = current.split("/");

            if (currentNums.length != 2) {
                System.out.println("Invalid input format. Please use the format (numerator/denominator).");
                i--;
                continue;
            }
            try {
                numerator[i] = Integer.parseInt(currentNums[0]);
                denominator[i] = Integer.parseInt(currentNums[1]);
                System.out.println("Read: " + numerator[i] + "/" + denominator[i]);
            } catch (NumberFormatException e) {
                System.out.println("Invalid numeric input. Please enter valid integers for numerator and
denominator.");
                i--;
            }
        }

        String average = getAverage(numerator, denominator, length);
        System.out.println("The average rational number is: " + average);

        String max = getMax(numerator, denominator, length);
        System.out.println("The max rational number is: " + max);

        String min = getMin(numerator, denominator, length);
        System.out.println("The min rational number is: " + min);

        sc.close();
    }

    public static String toString(int top, int bottom) {
        return top + "/" + bottom;
    }

    private static int greatestCommonDen(int top, int bottom) {
        if (top == 0)
            return bottom;
        return greatestCommonDen(bottom % top, top);
    }

    private static int lowestCommonDen(int top, int bottom) {
```

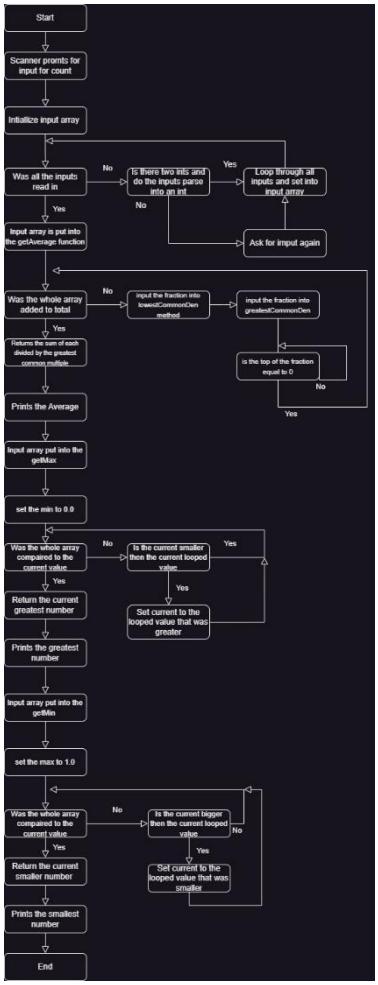
```
    return (top * bottom) / greatestCommonDen(top, bottom);
  }

  private static String getAverage(int[] numerator, int[] denominator, int count) {
    int commonDenominator = denominator[0] * count;
    for (int i = 1; i < denominator.length; i++) {
      commonDenominator = lowestCommonDen(commonDenominator, denominator[i] * count);
    }

    int totalSumNumerator = 0;
    for (int i = 0; i < numerator.length; i++) {
      totalSumNumerator += (numerator[i] * (commonDenominator / (denominator[i] * count)));
    }

    int greatestCommonDen = greatestCommonDen(totalSumNumerator, commonDenominator);
    return toString(totalSumNumerator / greatestCommonDen, commonDenominator /
greatestCommonDen);
  }
  private static String getMax(int[] numerator, int[] denominator, int count){
    double max = 0.0;
    int maxIndex = -1;
            for (int i = 0; i < numerator.length; i++) {
      double current = (double) numerator[i] / denominator[i];

      if (current > max) {
        max = current;
        maxIndex = i;
      }
    }
    return toString(numerator[maxIndex], denominator[maxIndex]);
  }
  private static String getMin(int[] numerator, int[] denominator, int count){
    double min = 1.0;
    int minIndex = -1;
            for (int i = 0; i < numerator.length; i++) {
      double current = (double) numerator[i] / denominator[i];

      if (current < min) {
        min = current;
        minIndex = i;
      }
    }
    return toString(numerator[minIndex], denominator[minIndex]);
  }
```

}

d) Compare the actual time spent to your estimates form Part (a) above.

Type of activity	Estimated Time	Actual Time
Understanding the problem	10 mins	10 mins
Identifying the structure	5 mins	5 mins
Coding	25 mins	10 mins
Testing (data set, compiling, error issues)	10 mins	10 mins
Documentation	25 mins	25 mins
Total time of project	**1 hour and 15 minutes**	**1 hour**

e) Code analysis part: Calculate McCabe's Cyclomatic Complexity for our actual entire program for this Exercise 4. Show your calculations in your assignment report.

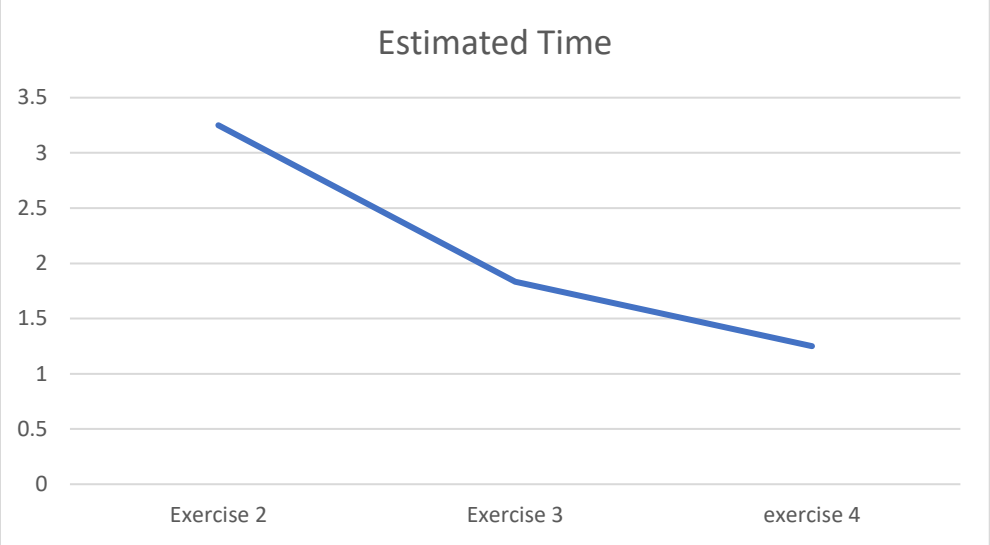


Cyclomatic complexity = E – N + 2p,
where E = number of edges of the graph,
N = number of nodes of the graph, and
p = number of connected components (usually p = 1)
E = 34
N = 29
P = 1

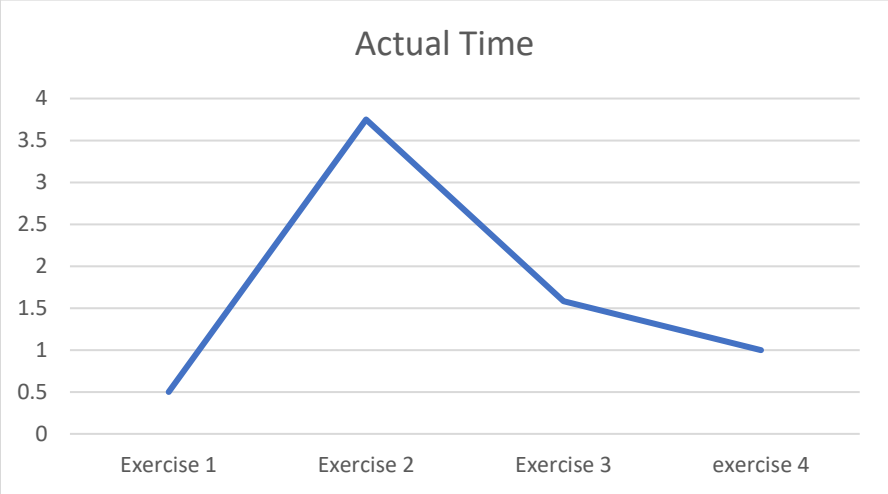
Cyclomatic complexity = 34 – 29 + 2(1)
Cyclomatic complexity = 7

Exercise 5

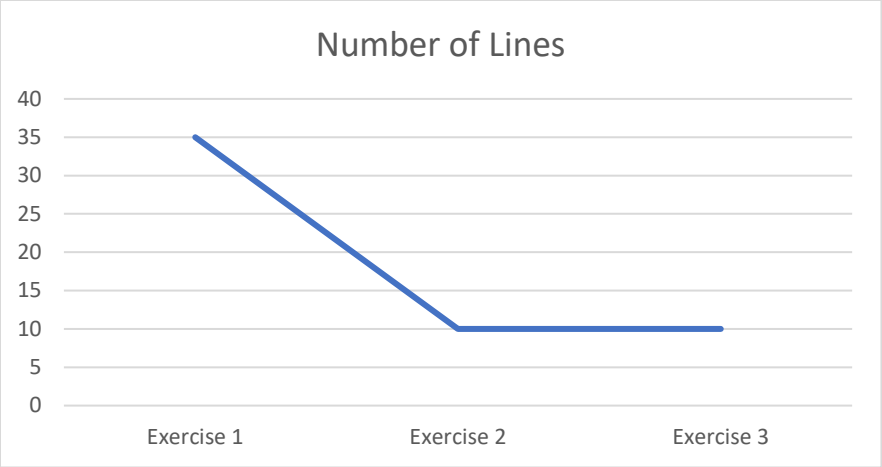
a) Show on a graph the total effort estimates for Exercises 2, 3, 4.



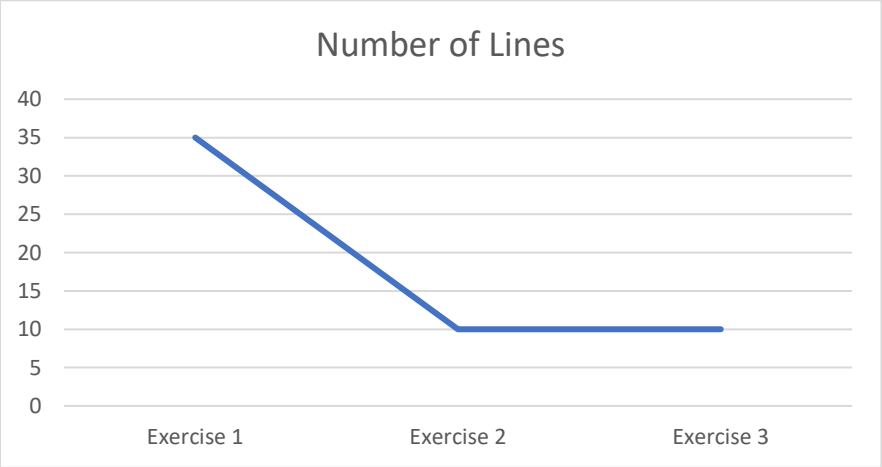
b) Show on a graph the total actual effort values for Exercises 1, 2, 3, 4 (include the effort value form Exercise 1).



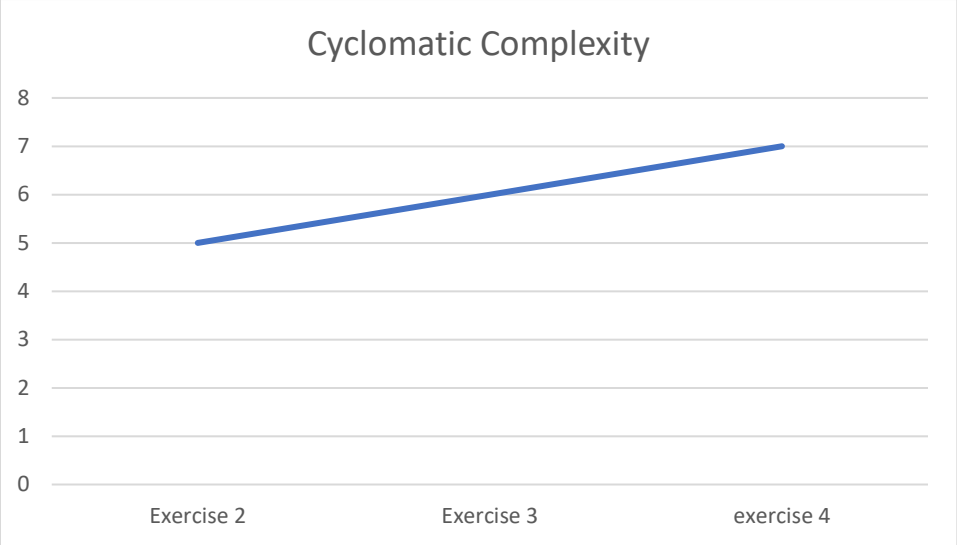
c) Show on a graph the count for the total lines of code in each of Exercises 2, 3, 4. Count every line in your source code files (you may use *wc* command in Linux).



d) Show on a graph the count for the total lines of code in each of Exercises 1, 2, 3. Count every line in your source code files (you may use wc command in Linux).

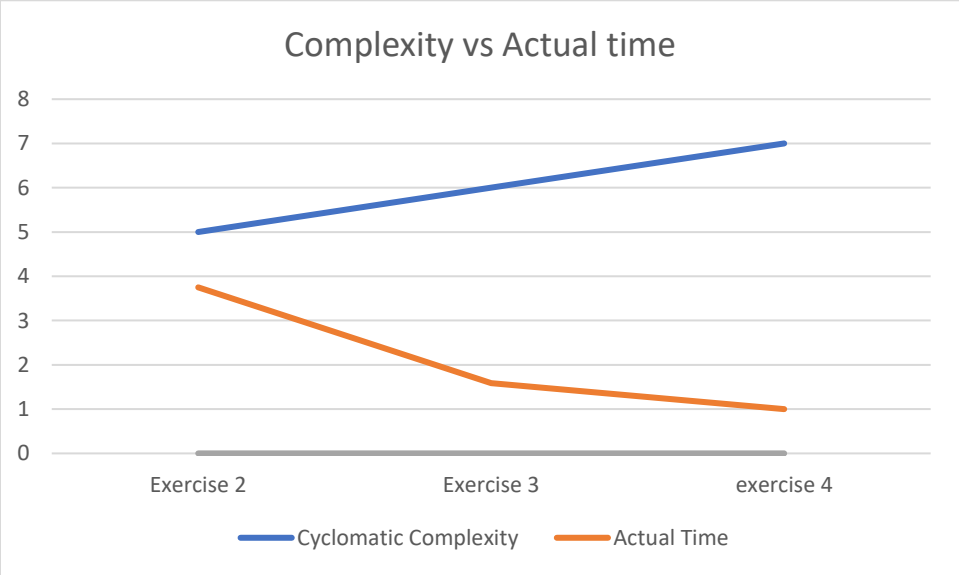


e) Show on a graph the values calculated using McCabe's Cyclomatic Complexity for programs in Exercises 2, 3, 4.



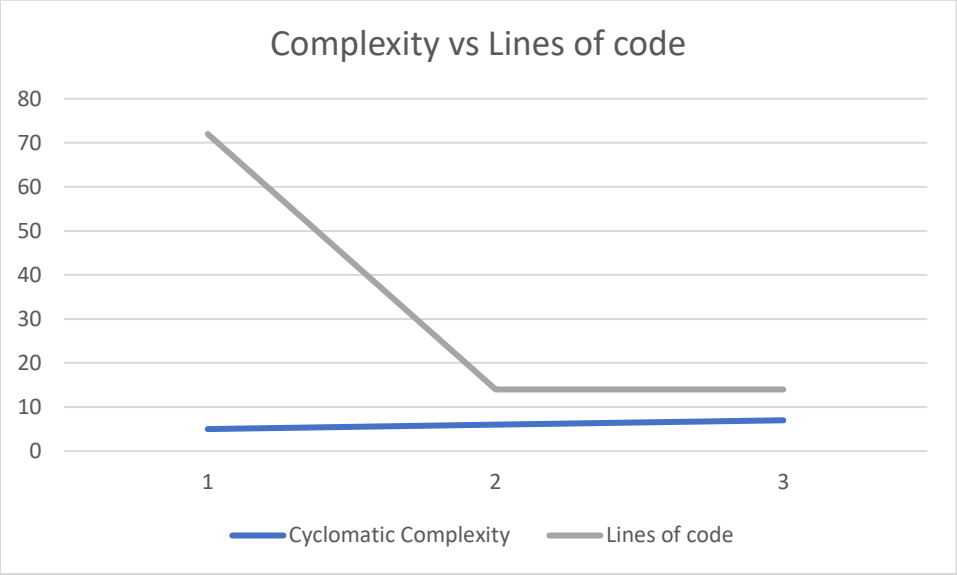
Exercise 6

When looking at assignment 1 compared to this assignment we see the same trend as this. complexity goes up and our actual time effort goes down. Why, probably due to code reusability, being able to copy the code I wrote to get the greatest number and paste the code for the get the smallest number method, with only having to change the if operator from greater the to less than. (Figure 5.1)



(Figure 5.1)

The growth rate is the same as assignment 1 when comparing graphs, as well. For exercise 2 I wrote much more code compared to exercise 4, but the code was much simpler than exercise 4. For exercise 2 most of the code was setting up my scanner to take in the inputs rather than the actual average number method. As it goes on there is less code, but it gets more complex with less actual written code. (Figure 5.2)



(Figure 5.2)