Code Effiency Report

COS 750 - Software Engineering

Thursday, November 7, 2024

Hi Kuda C

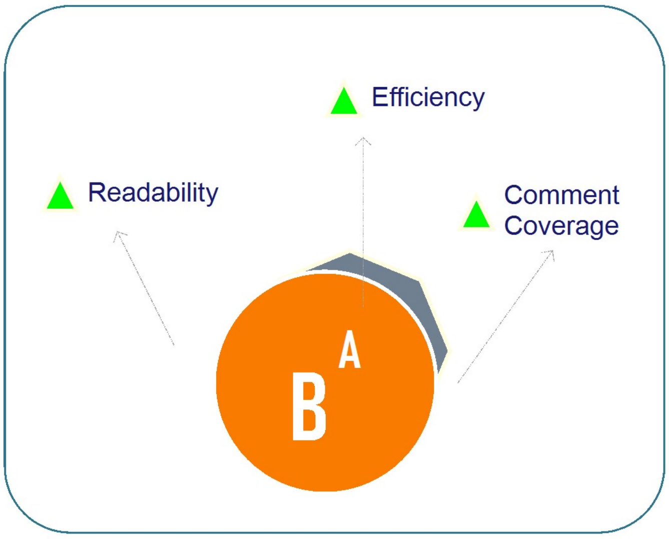
Kuda C Chivunga u19068752

Awesome work on your assignment! Dive into the report below to see how your code performed. Keep pushing those coding skills to the next level!

Execution Time: 22ms CPU Usage: 0.5s

Memory Usage: 23678 bytes

Cyclomatic Complexity -	4 -
Code Coverage	: 75%
Mantainabilty Index	:85%

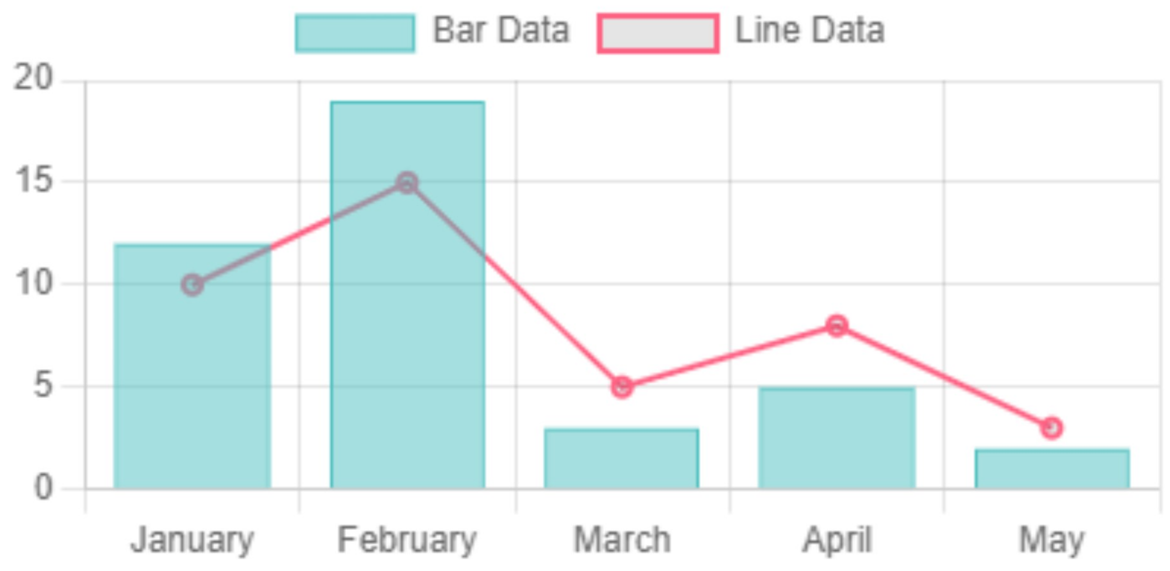


Overall Efficiency Score Over Time

Performance OverView

Assignment	Topic	Score	Average
Assignment 1	Arrays and Lists	12	10
Assignment 2	Stacks and Queues	19	15
Assignment 3	Trees (Binary and AVL)	3	5
Assignment 4	Graphs (DFS and BFS)	5	8

This graph illustrates your overall efficiency score across different assignments. Notice the upward trend, indicating consistent improvement in your coding skills and efficiency. Keep up the good work!



19	2	4
Lines Of Code	*Number Of Functions*	*Number Of Comments*

Detailed Effieciency Break Down

This page provides a comprehensive analysis of your code quality. It compares your submitted code with industry standards, offering alternative code suggestions to enhance performance. You'll find explanations of key metrics and their impact on your code quality, along with actionable feedback to help you improve. Additionally, this section highlights your progress over time by comparing current metrics with past assignments.

Duplicate Code:

No Duplicate Code Detected

Readability

1. Comment Coverage

Score: 75%

Why Comment? Proper use of comments is crucial for improving code readability and understanding. Comments should be used to explain the logic behind complex code segments or algorithms, making it easier for others to maintain and modify the code. In this submission, while some sections are well-commented, there are areas where additional explanations could enhance clarity, especially around complex loops and conditionals. Consider adding comments to these sections to improve overall understanding and ensure maintainability.

1.Naming of Variables

Score: 99%

Why? Clear and descriptive variable names are crucial for maintaining readable and understandable code. In this submission, some variable names are generic (e.g., a, b, temp), which makes it harder to understand their purpose within the code. For instance, variables related to holding specific values (e.g., sum of elements, counter for loops) should have more descriptive names like currentSum, elementCounter, or maxValue to provide more context.

Complexity

Code is well structured and readable - how it

1. Cyclomatic Complexity

Score: 4 - Low Complexity

Cyclomatic Complexity is a metric used to measure the complexity of a program by quantifying the number of linearly independent paths through the program's source code. It was introduced by Thomas McCabe in 1976 as a way to gauge the structural complexity of a program and has since become a fundamental software metric for assessing code quality.

Cyclomatic Complexity is calculated based on the control flow graph of the program, where each node represents a block of code, and each edge represents a control flow path. The formula to compute the cyclomatic complexity

$V(G) = E - N + 2p$

1.Maintanability Index

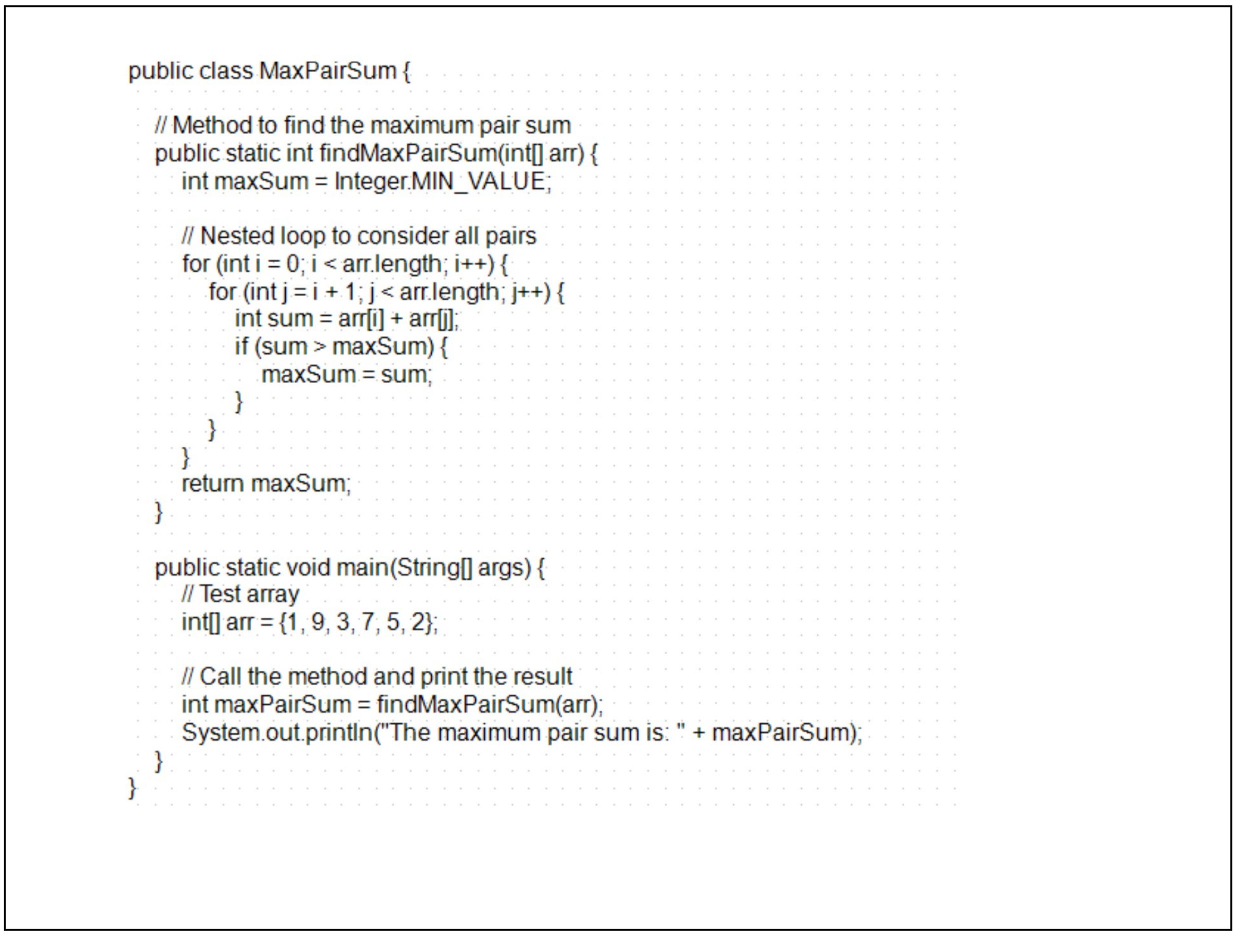
Score:1.386 - High Maintanability Index

This indicates that the code is in a good state.The Maintainability Index is a composite measure used to evaluate how maintainable a piece of software is. Higher values suggest more maintainable code. The scale typically goes from 0 to 100, with a higher score indicating that the code is easier to maintain, refactor, and extend.

1.Time Complexity

Score:1.386 - High Maintanability Index

This loop runs from $i = 0$ to $i < n$ where n is the length of the array. This gives the outer loop a time complexity of O(n).For each iteration of the outer loop, the middle loop runs from $j = i + 1$ to $j < n$.
This means the number of iterations decreases with each increment of
i. The total number of iterations for this loop is proportional to n, but for each iteration of i, it executes n-1 times.



```java
public class MaxPairSum {

    // Method to find the maximum pair sum
    public static int findMaxPairSum(int[] arr) {
        int maxSum = Integer.MIN_VALUE;

        // Nested loop to consider all pairs
        for (int i = 0; i < arr.length; i++) {
            for (int j = i + 1; j < arr.length; j++) {
                int sum = arr[i] + arr[j];
                if (sum > maxSum) {
                    maxSum = sum;
                }
            }
        }
        return maxSum;
    }

    public static void main(String[] args) {
        // Test array
        int[] arr = {1, 9, 3, 7, 5, 2};

        // Call the method and print the result
        int maxPairSum = findMaxPairSum(arr);
        System.out.println("The maximum pair sum is: " + maxPairSum);
    }
}
```