Course Code: CSE 205 Course Title: Algorithms Theory

KSA – 1 : Assignment

**Assignment No.** 01

**Student Name :** Jahidul Islam

**Student ID** 221002504

**Section :** 221-D7 **Date of Assignment :** 6/12/2023 **Date of Submission :** 24/12/2023

**Submitted to : Dr. Faiz Al Faisal,** Assistant Professor & Associate Chairperson, CSE, Green University of Bangladesh

## To be Filled by the Teacher.

**Received Date :**

**Late Submission :**

**Obtained Mark :**

**Remarks :**

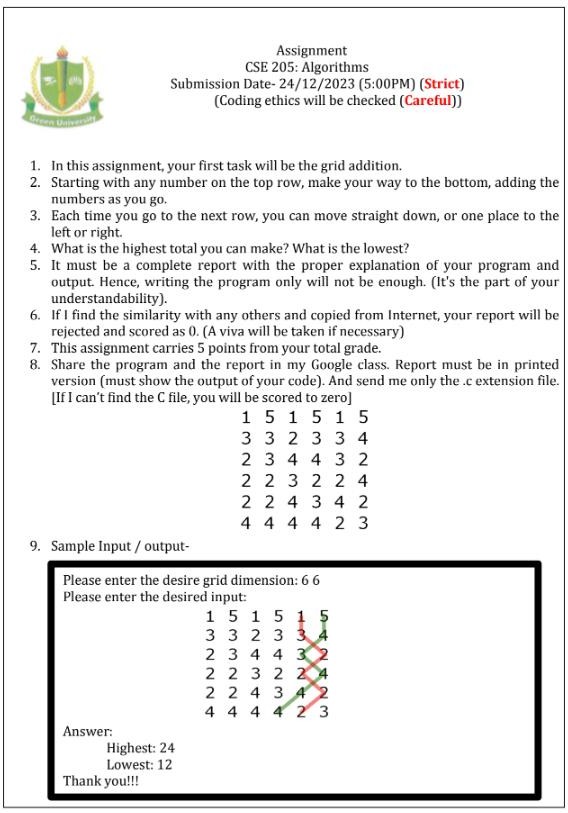
**Signature:**



**Department of Computer Science Engineering**

**Green University of Bangladesh**

# TITLE:



1. **PROBLEM STATEMENT:**

The grid addition problem involves finding the highest and lowest totals while traversing a grid from the top row to the bottom row. The goal is to determine the maximum and minimum sums achievable by moving down to adjacent cells in the subsequent rows. Each move can be either straight down or one place to the left or right.

# OBJECTIVES/AIM:

Objective 1: Develop a dynamic programming solution for the Grid problem.

Objective 2: Provide flexibility for users to input different coin denominations. By changing the array of coins.

Objective 4: Output the highest and lowest totals

# PROCEDURE:

The grid addition problem is solved using a dynamic programming approach in Java.

* 1. **Initialization:**

grid = {

{1, 5, 1, 5, 1, 5},

{3, 3, 2, 3, 3, 4},

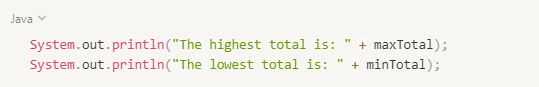
{2, 3, 4, 4, 3, 2},

{2, 2, 3, 2, 2, 4},

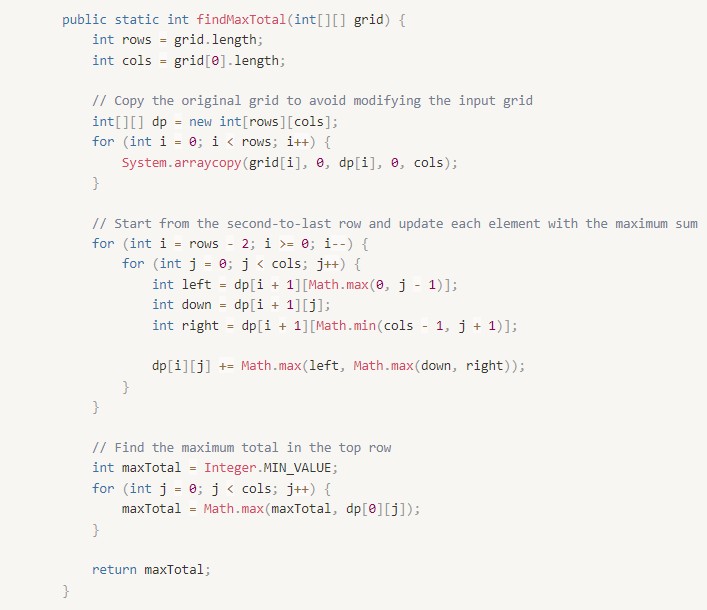
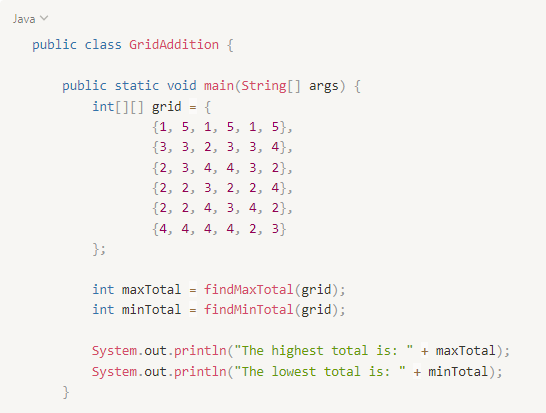
{2, 2, 4, 3, 4, 2},

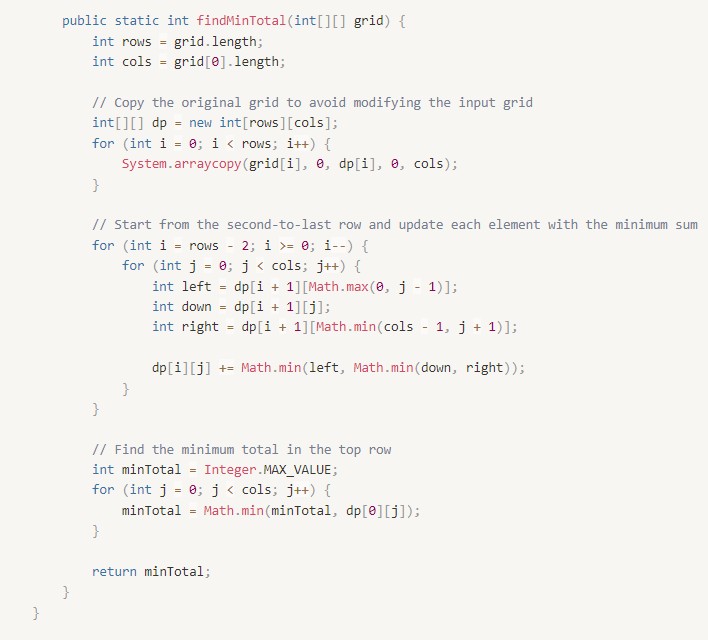
{4, 4, 4, 4, 2, 3}

};

* 1. **Dynamic Programming - Fill the Matrices:**
     + Implement the dynamic programming approach to find the highest and lowest totals:
       - Utilize two separate functions, **findMaxTotal** and **findMinTotal**, to calculate the maximum and minimum totals, respectively.
       - For each function:
         * Create a copy of the original grid to prevent modification of the input.
         * Start from the second-to-last row and iterate upwards, updating each element based on the maximum or minimum sum of the available options (straight down, left, or right).
         * Calculate the final totals considering the possibilities at each step.
  2. **User-Friendly Input:**
     + For simplicity, the program currently uses predefined Grid. Users can customize this array as needed.
  3. **Output:**
     + Finally, the program Print the highest and lowest totals: to the console.

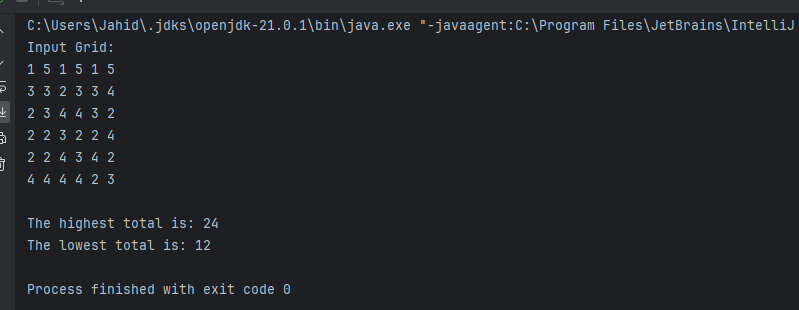
# IMPLEMENTATION: CODE IN JAVA:

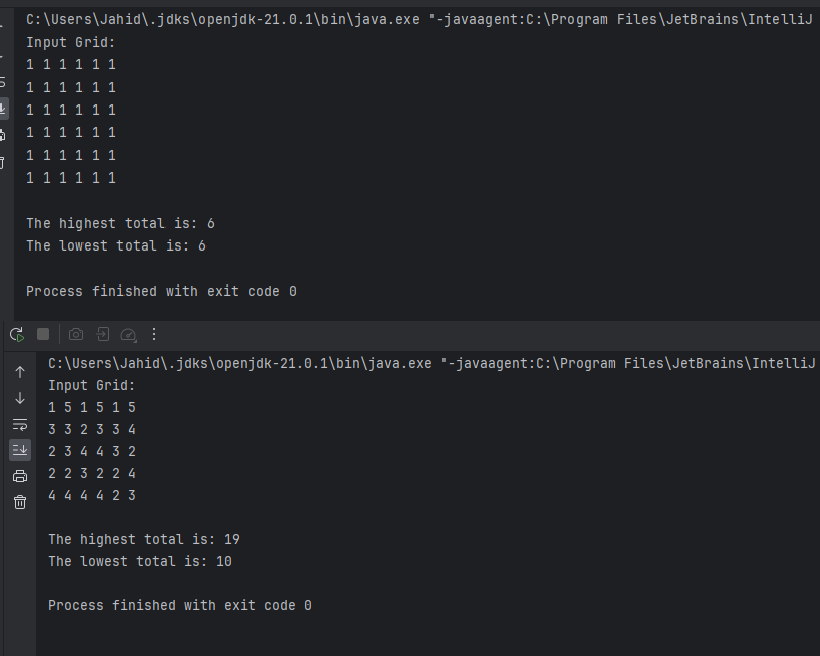




1. **TEST RESULT / OUTPUT:**

* Objective 1:





# DISCUSSION AND ANALYSIS:

## Dynamic Programming Solution:

The implementation employs a dynamic programming approach to efficiently calculate the Max and Min value. This ensures optimal time complexity by avoiding redundant calculations and storing intermediate results in the dp array.

## User Input Flexibility:

The program offers flexibility by allowing users to input the target amount and customize the coin denominations. This makes the solution adaptable to various scenarios where different denominations may be required.

## Efficiency:

The algorithm's efficiency is notable, particularly for larger target amounts, as the dynamic programming technique optimizes the computation by breaking down the problem into smaller subproblems and reusing solutions.

## Scalability:

The solution is scalable, and additional problems like solving Maze, finding path in maze can be easily incorporated.

# CONCLUSION:

* The procedure involves initializing the input, applying a dynamic programming approach through separate functions, displaying the results, and providing insights into the time complexity.
* The program successfully addresses the grid addition problem, yielding the highest and lowest totals attainable through the defined rules.

This procedure outlines the steps taken to execute the program, emphasizing clarity in input handling, dynamic programming, and results presentation.