Raj Oli

**BSc. Hons Computing,  Softwarica College of IT and E-commerce**

  ST5008CEM Programming for Developers

  Hikmat Saud

11 August,2023

**Github link**: <https://github.com/2025raj/progfordev.git>

**# Question Number 1**

a)

A trio of friends planned to purchase clothing from a particular store for an upcoming party, intending to wear matching outfits in varying colors - black, blue, and pink. The store had three different clothing sets on display, each in a different color. The shopkeeper assisted the three friends by selecting a clothing set in the appropriate color for each person based on their body shape and size. Given a 2D array, price[][3], where price[i][0], price[i][1], and price[i][2] represent the price of each clothing set for a different colored outfit for person i, your objective is to determine the minimum cost required to purchase clothing such that each person wears have different color clothes if they stand in a row. It should be noted that any two people can wear the same color cloth, but the third person must wear various color cloths, and all three can wear different colored garments.

Input: N = 3, price[][3] = [{14, 4, 11}, {11, 14, 3}, {14, 2, 10}]

Output: 9

Explanation:

Person 1 chooses blue clothing cost=4. Person 2 chooses pink clothing. Cost = 3. Person 3 chooses blue

clothing. Cost = 2.

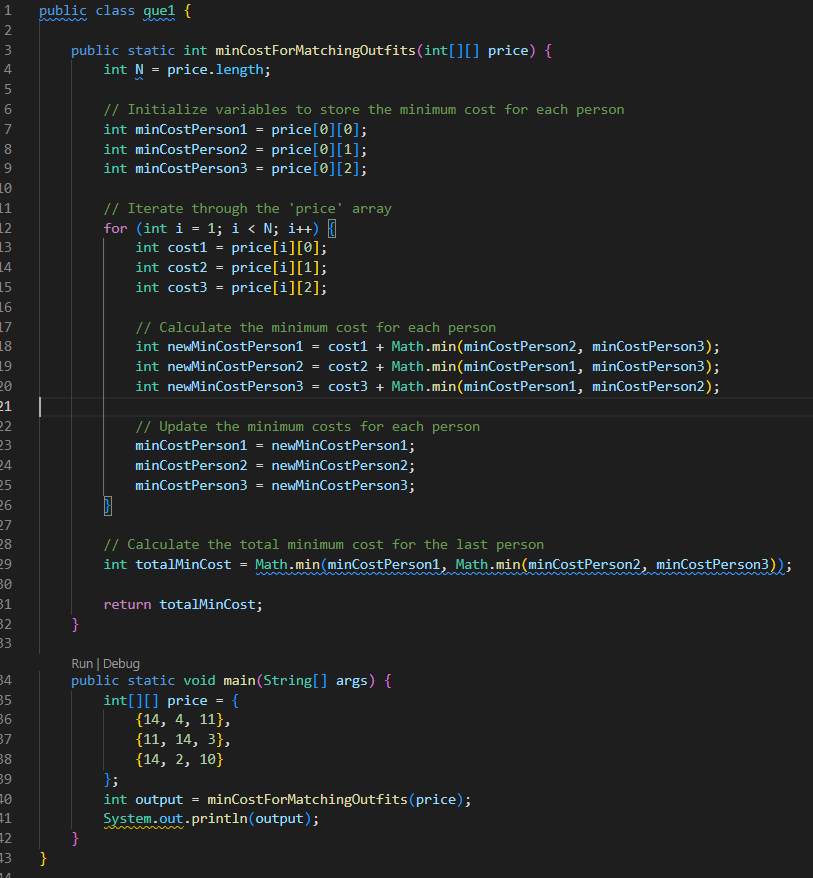
As a result, the total cost = 2 + 5 + 3 = 9.

Note: algorithm must take

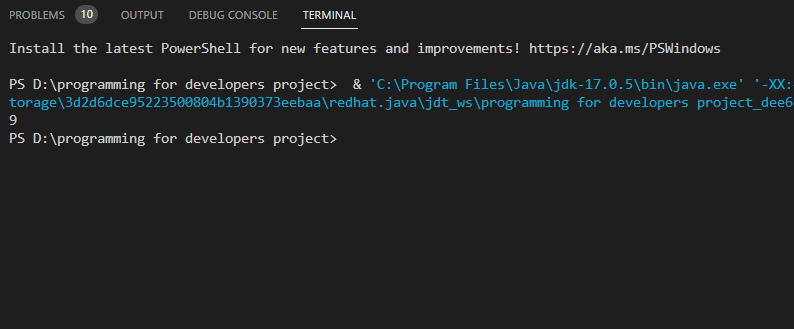
Time Complexity: O(N)

Auxiliary Space: O(1)

**Code;**



**Output ;**



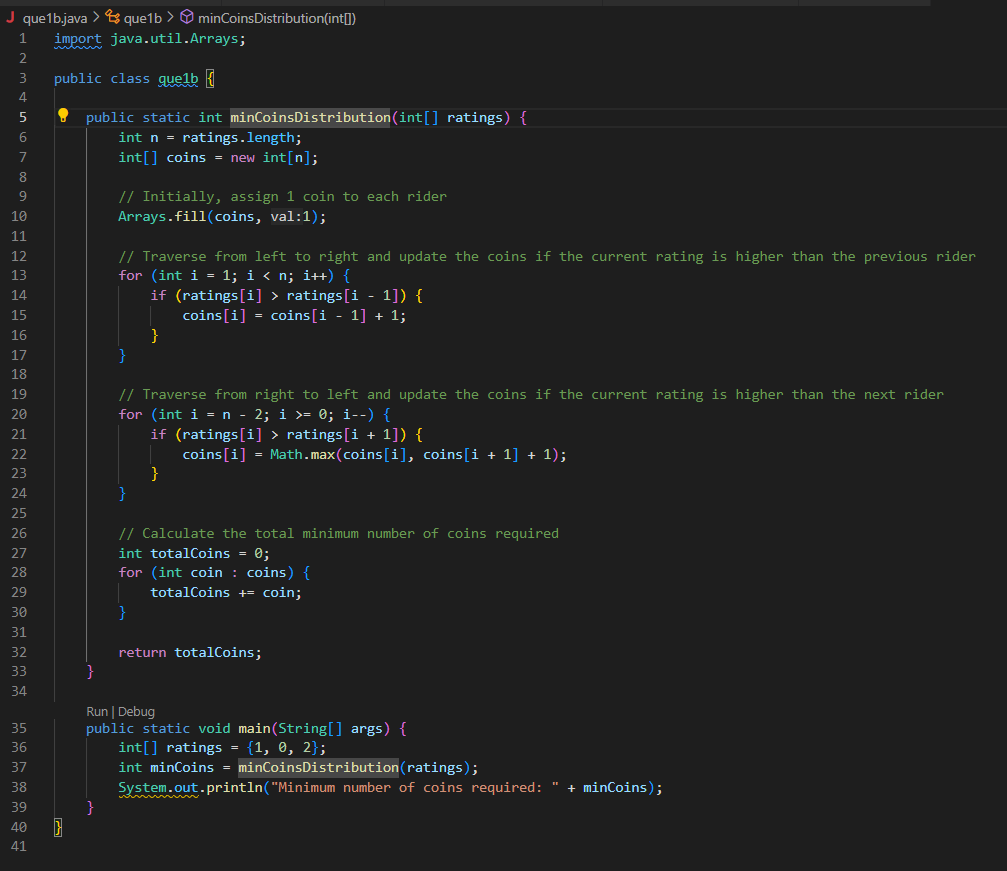
**1.b**

A group of n Pathao riders stood in a queue, and each rider was assigned a unique integer rating based on their performance over the year. The Pathao company planned to distribute gold coins to each rider in ascending order, starting from count 1. The riders with higher ratings should receive more coins than their neighboring riders. The objective was to determine the minimum number of coins required by Pathao to distribute coins to the selected riders according to their ratings.

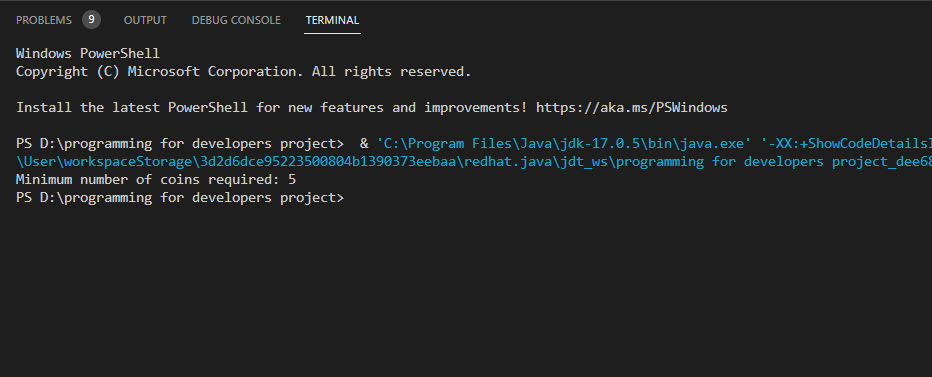
Input: ratings = [1,0,2]

Output: 5

Explanation: You can give the first, second, and third rider 2, 1, 2 gold coins, respectively.

Code;

**Output;**



**# Question Number 2**

1. a) Given an integer array nums and another integer k, the goal is to find the longest subsequence of nums that satisfies the following two conditions:

The subsequence is strictly decreasing.

The difference between adjacent elements in the subsequence is at most k.

The output should be the length of the longest subsequence that meets these requirements.

For example, consider the following input:

nums = [8,5,4, 2, 1, 4, 3, 4, 3, 1, 15] k = 3

output=[8,5,4,2,1] or [8,5,4,3,1]

Output: 5

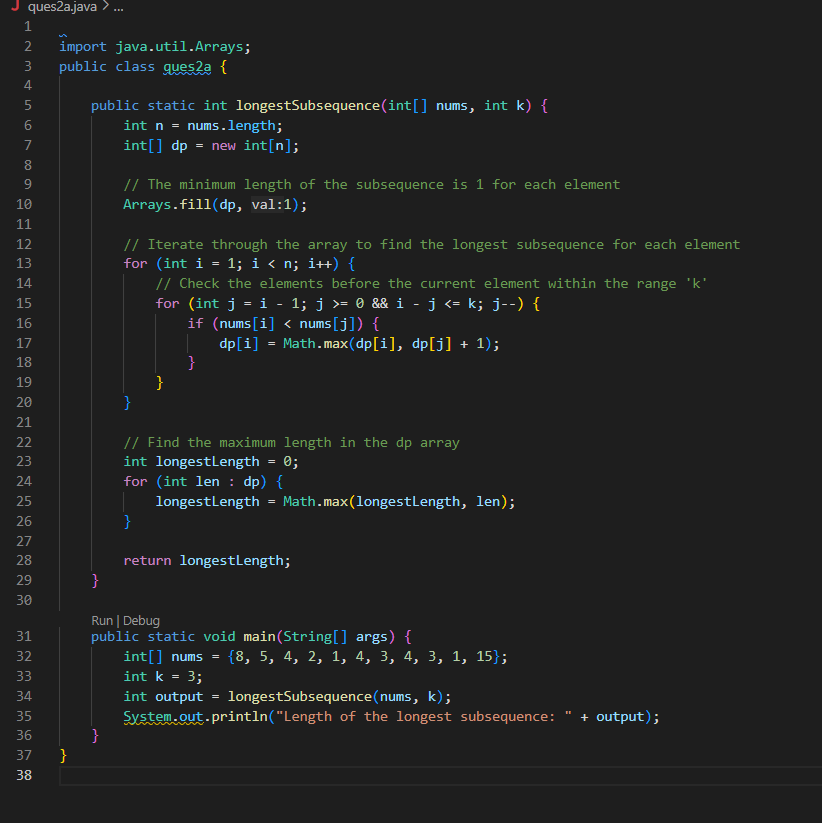
Explanation:

The longest subsequence that meets the requirements is [8,5,4,2,1] or [8,5,4,3,1].

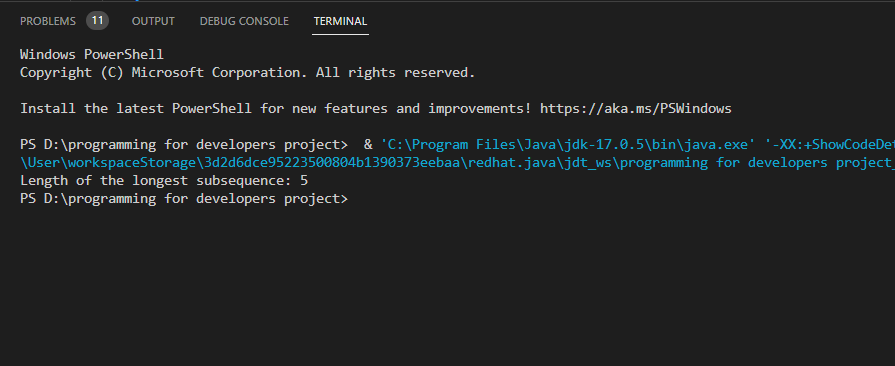
The subsequence has a length of 5, so we return 5.

Note that the subsequence [1,3,4,5,8,15] does not meet the requirements because 15 - 8 = 7 is larger than 3

**Code;**



**Output;**



**2.b**

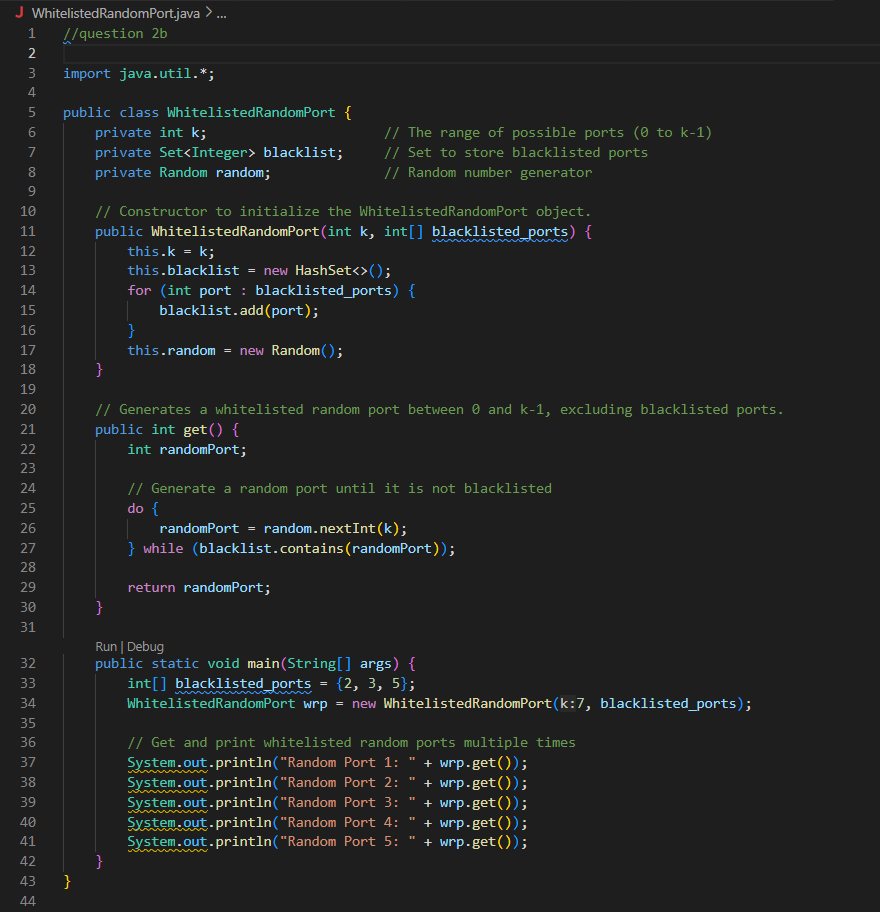
Given an integer value k and an array of integers representing blacklisted ports, create an algorithm that outputs a random port (an integer value between 0 and k-1) that is also a whitelisted port (meaning it is not in the array of blacklisted ports). The goal is to minimize the number of built-in random calls in the algorithm. The program should have two inputs: k, an integer value, and blacklisted\_ports, an array of integers. The program should also have a get() function that returns a whitelisted random number between 0 and k-1. The algorithm should be optimized to reduce the number of built-in random calls required.

Example 1: Input ["Program", "get", "get "get", "get", "get"] [[7, [2, 3, 5]], [], [], [], [], [], [], []] Output [null, 0, 4, 6,1,4]

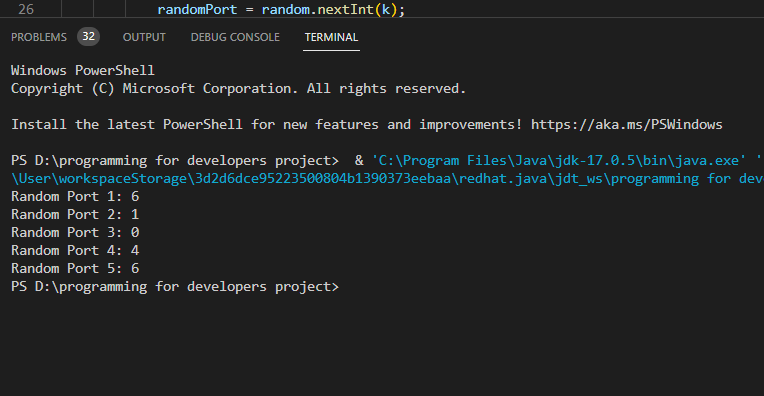
Explanation program p = new program(7, [2, 3, 5]); p.get(); // return 0, any number from [0,1,4,6] should be ok.

Note that for every call of pick, // 0, 1, 4, and 6 must be equally likely to be returned (i.e., with probability 1/4). p.get(); // return 4

**Code;**



**Output;**



**# Question Number 3**

1. Suppose you are provided an array of n targets that are placed in a row from 0 to n-1. Each target is assigned with certain integer such that a [0] represent the value associated with target zero. You are asked to shoot all the targets. If you shoot I th target then you will get a[i-1]\*a[i]\*a[i+1] points. Note that if i-1 and i+1 position hits index out of bound, then you can assume that two target with value 1 are padded before start target and after end target. Return maximum point one can gain by hitting each target.

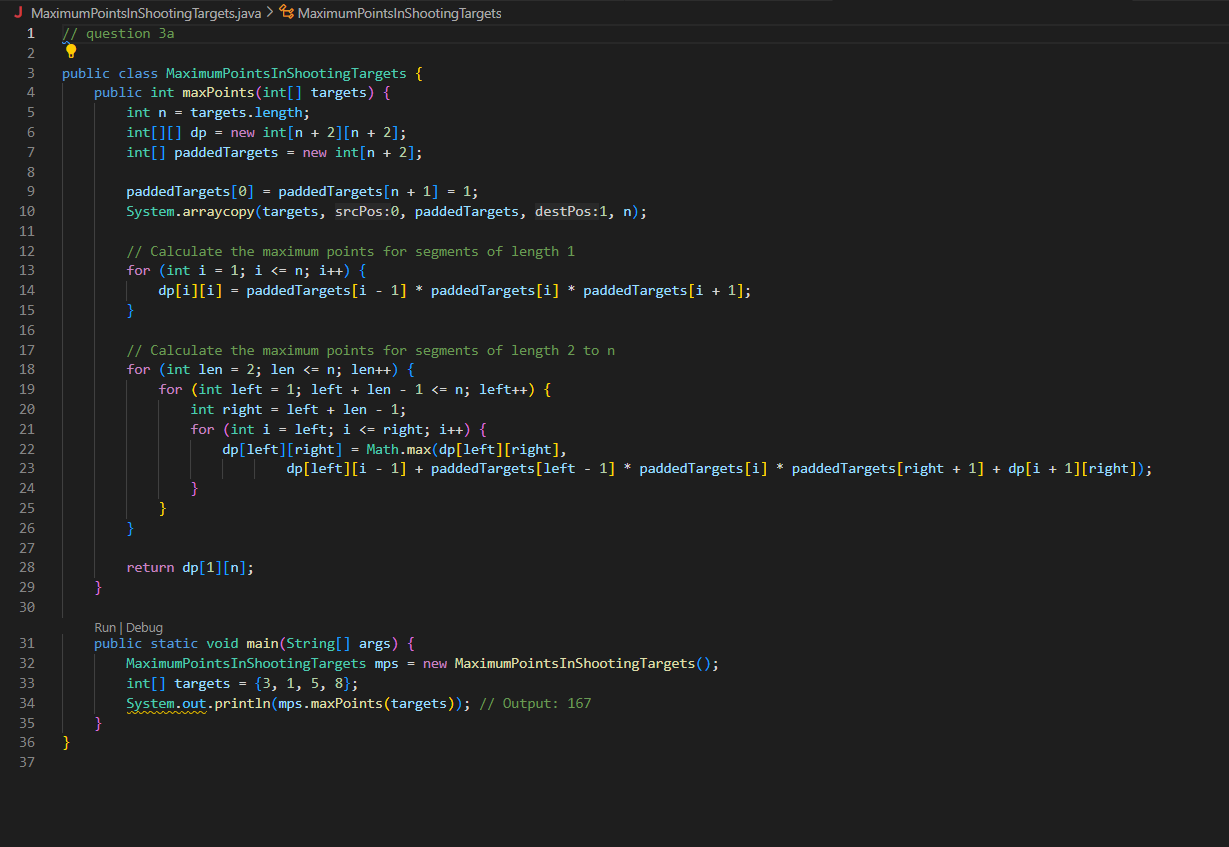
Input: a = [3,1,5,8]

Output: 167

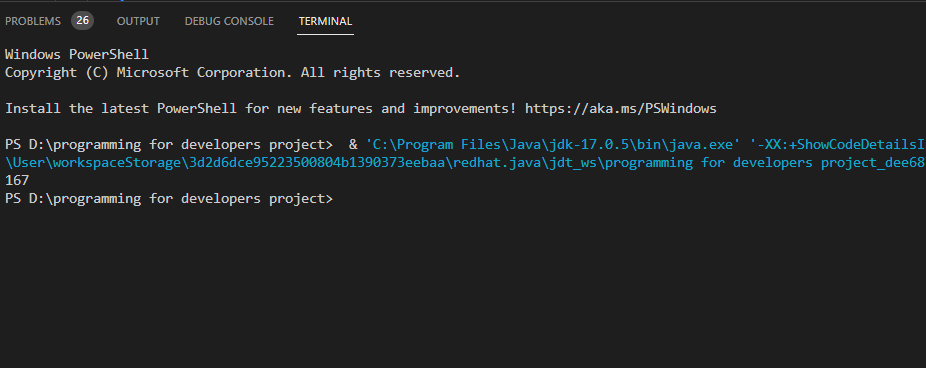
Explanation: a = [3,1,5,8] [3,1,5,8] points 3\*1\*5 (“hitting target with value 1”) [3,5,8] points 3\*5\*8 (“hitting target with value 5”) [3,8] points 1\*3\*8 (“hitting target with value 3”)

note that there is padded target with value 1 at beginning and end of the provided target list ,[8] points 1\*8\*1 same as above points = 3\*1\*5+ 3\*5\*8 + 1\*3\*8 + 1\*8\*1 = 167

**Code;**

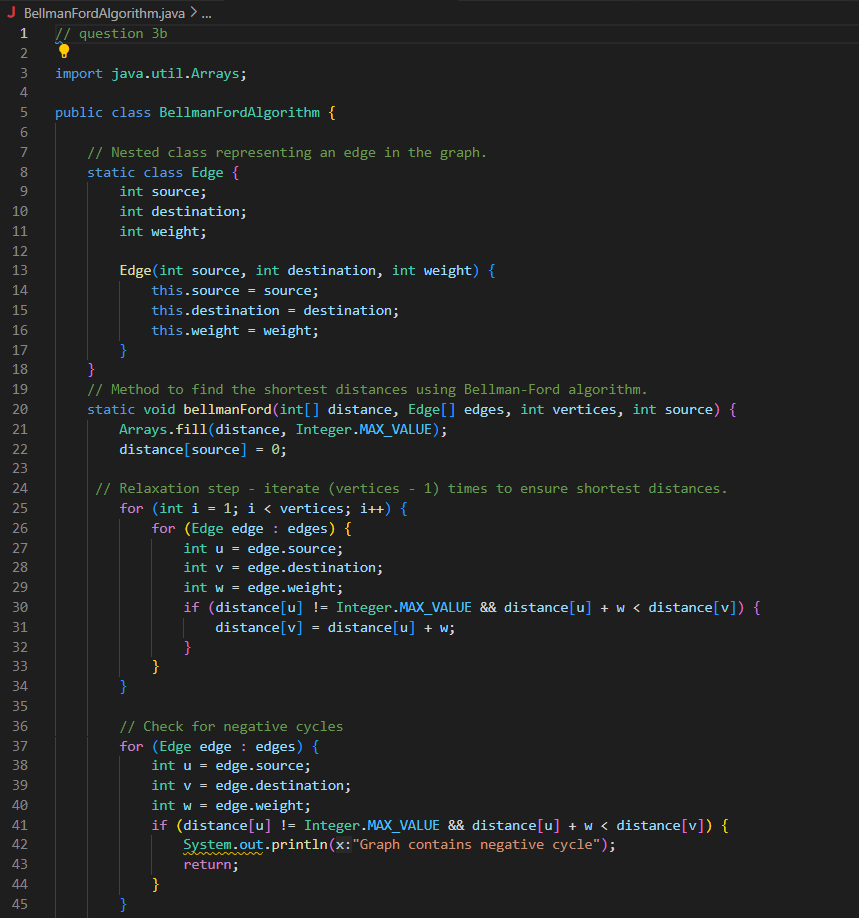


**Output;**



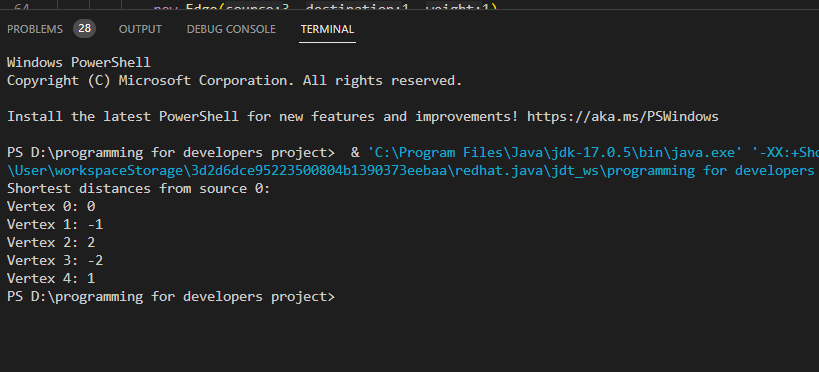
**3.b**

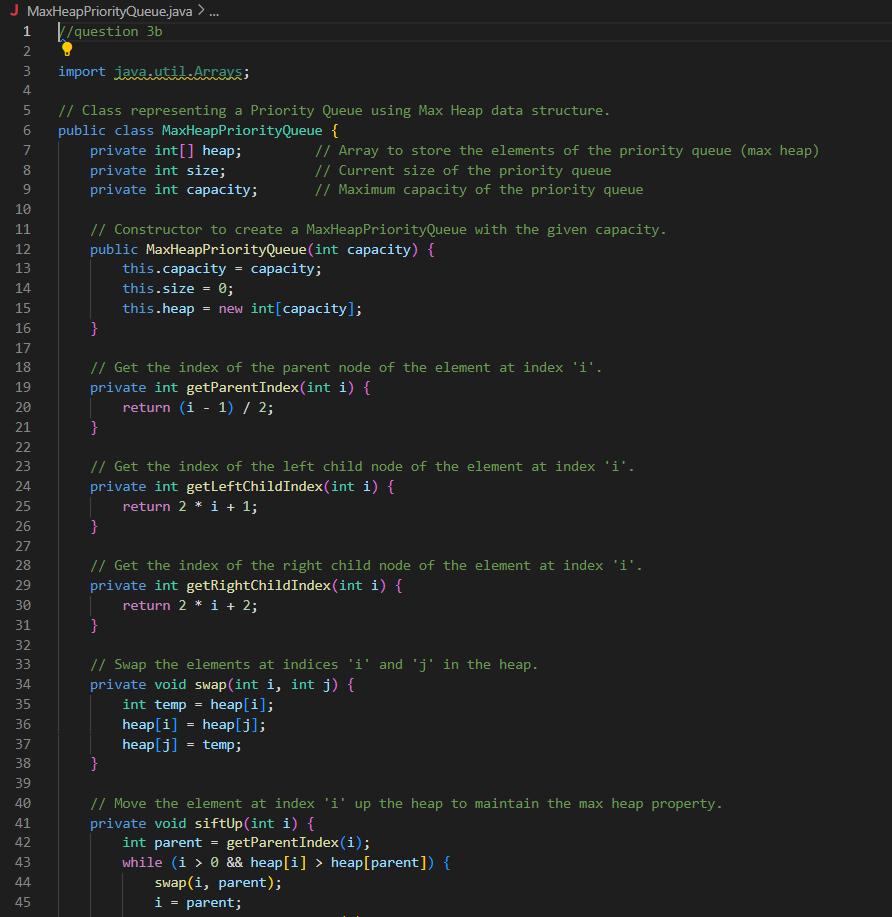
Implement bellman ford algorithm and priority queue using maximum heap.

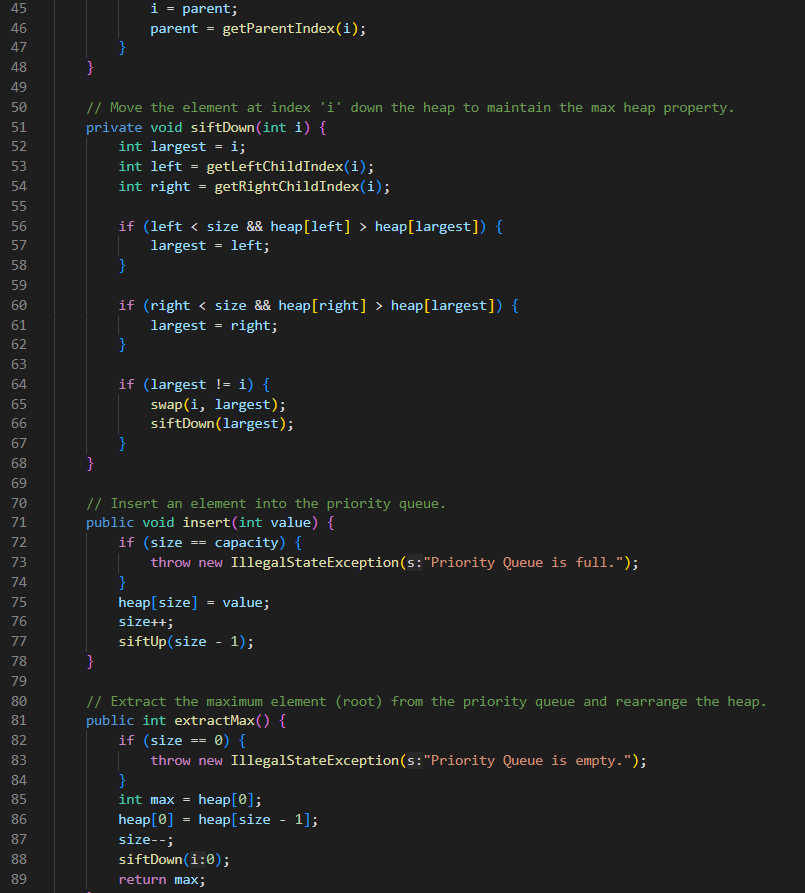
**Code**

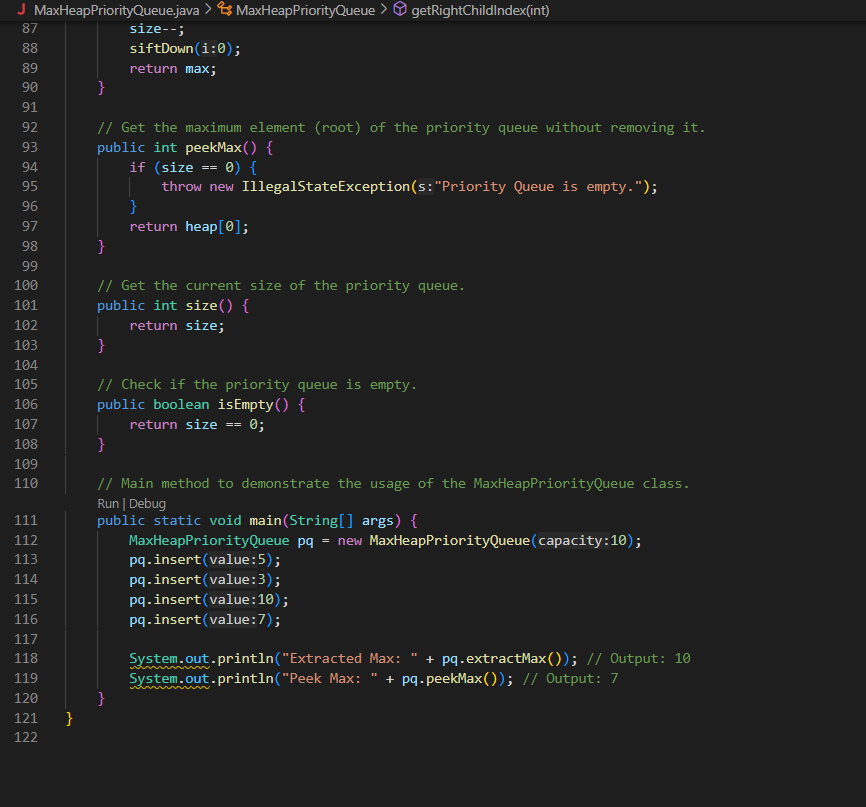


**Output;**

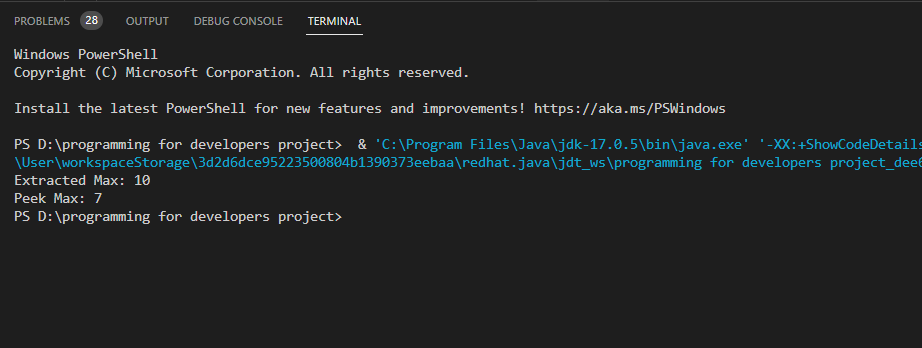








**Output;**



**# Question 4**

4a.

There are n tasks you need to complete for a game, labelled from 1 to n.

We are given r[i]=[x,y] representing a prerequisite relationship between task x and task y: task x has to be completed before task y.

In one step you can complete any number of task as long as you have completed all the prerequisites for the tasks you are provided while playing game.

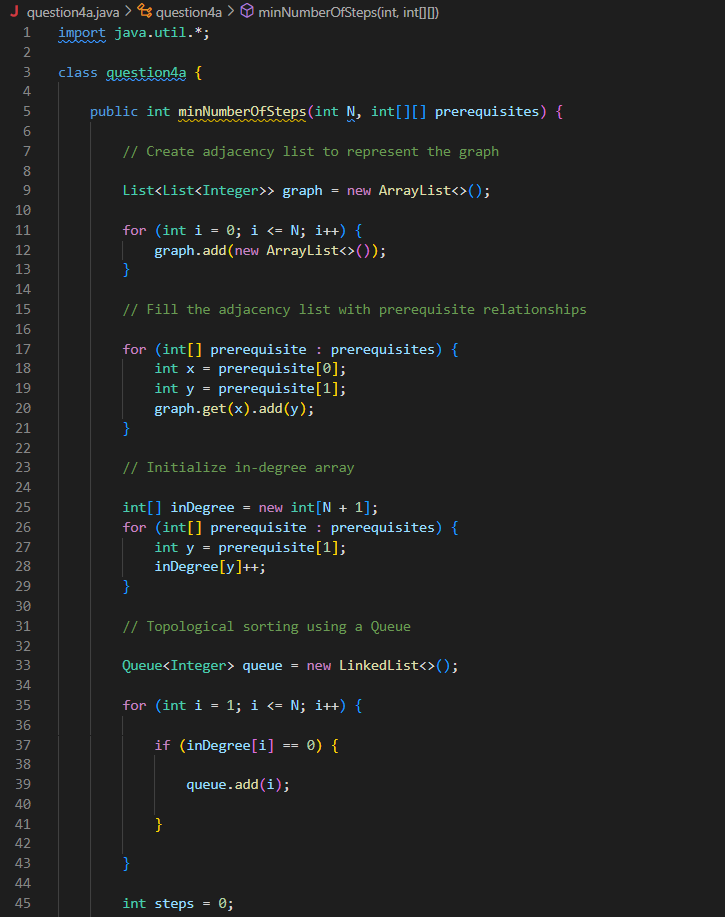
Return the minimum number of steps needed to complete all tasks. If there is no way to complete all the tasks, return -1.

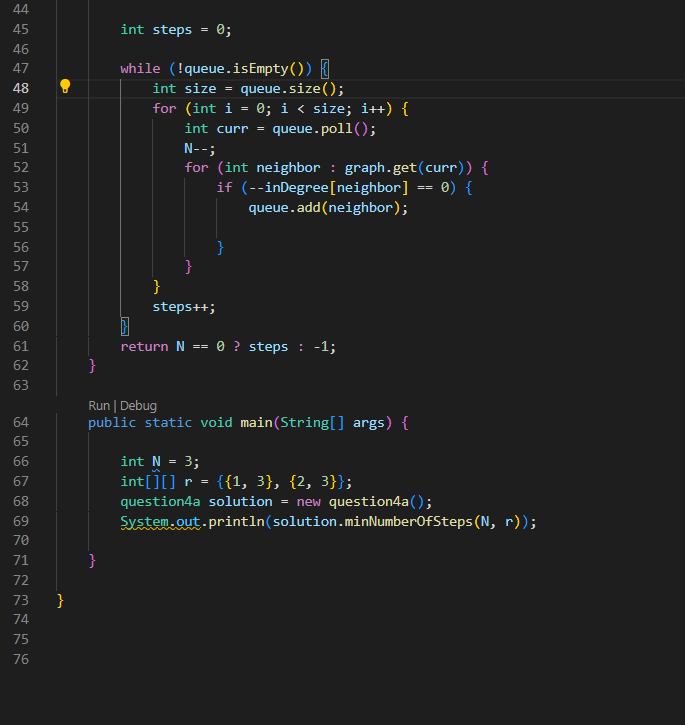
Input: N = 3, r= [[1,3],[2,3]]

Output: 2 Explanation: In the first step, you can complete task 1 and 2.

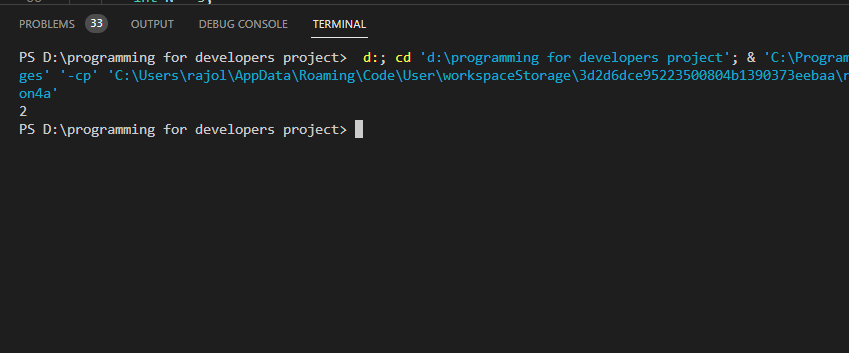
In the second semester, step task 3 can be completed.

**Code;**





**Output;**



1. Given the root of a binary tree with unique values and the values of two different nodes of the tree x and y, return true if the nodes corresponding to the values x and y in the tree are brothers, or false otherwise.

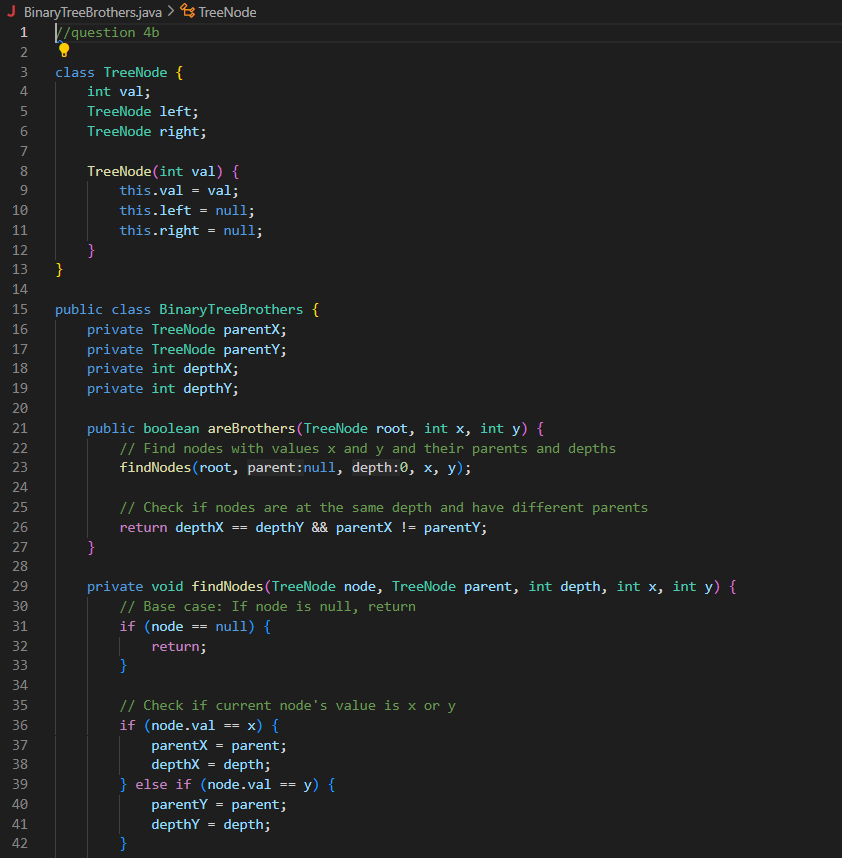
Two nodes of a binary tree are brothers if they have the same depth with different parents.

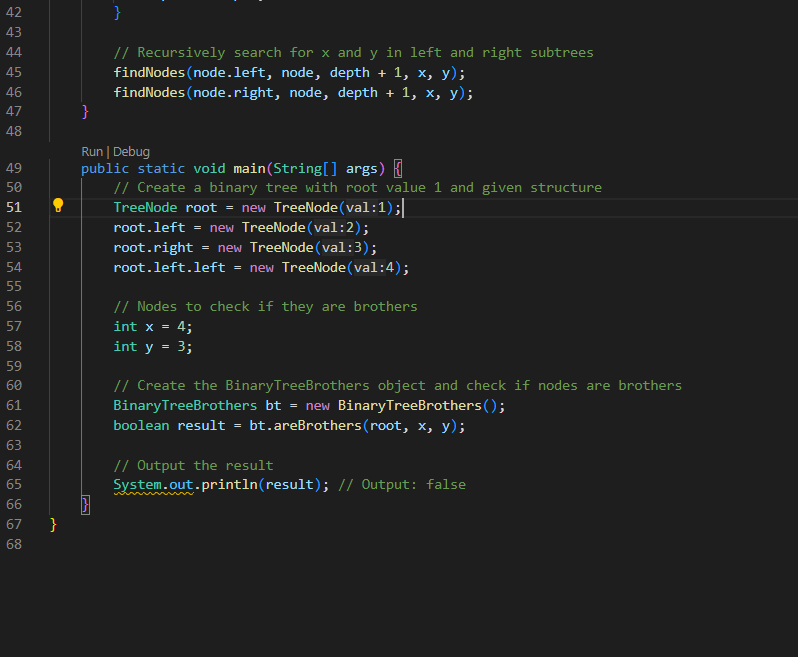
Note that in a binary tree, the root node is at the depth 0, and children of each depth k node are at the depth k + 1.

Input: root = [1,2,3,4], x = 4, y = 3

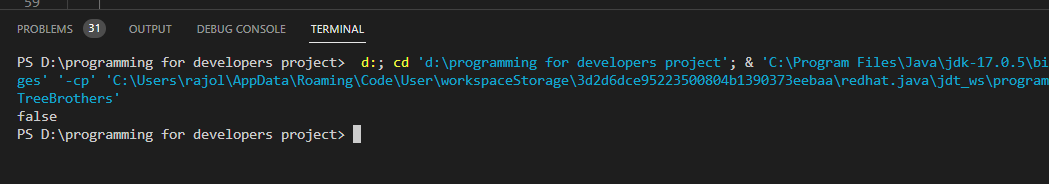
Output: false

**Code;**





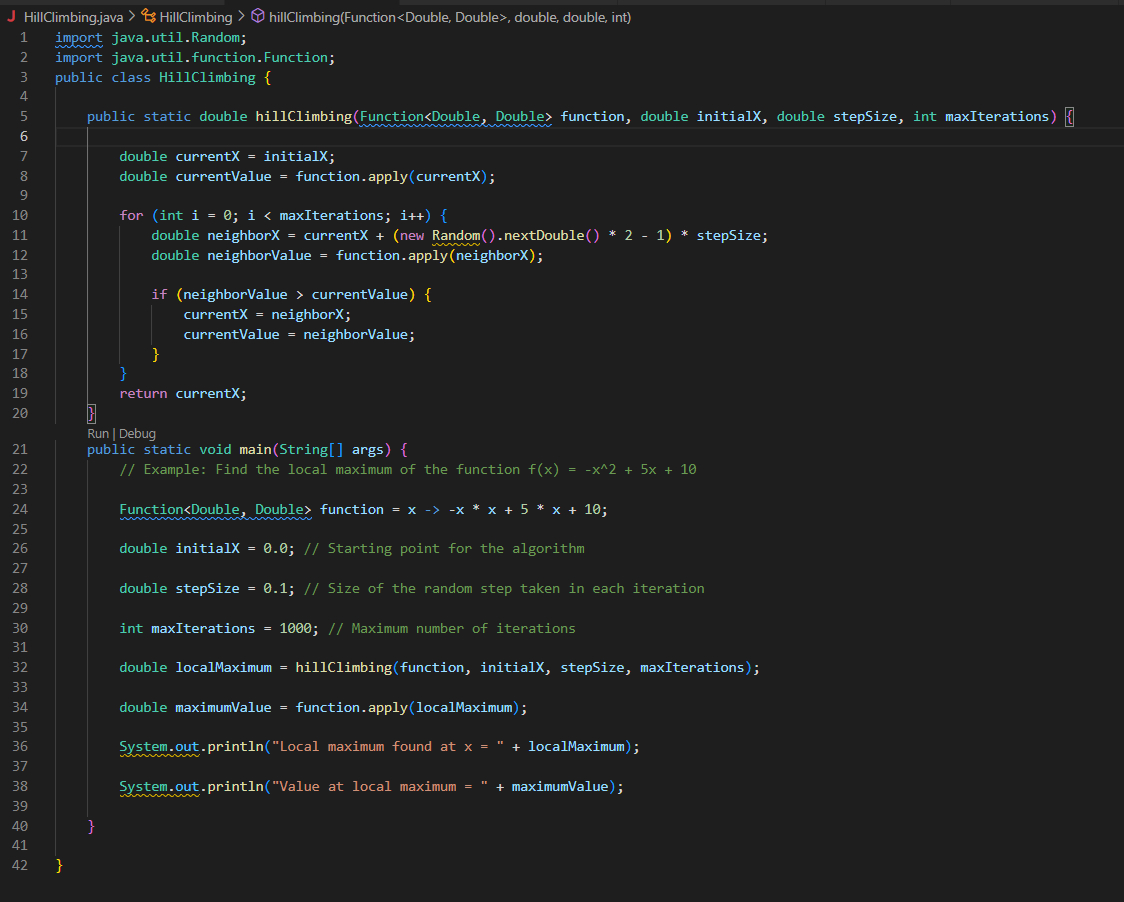
**Output;**



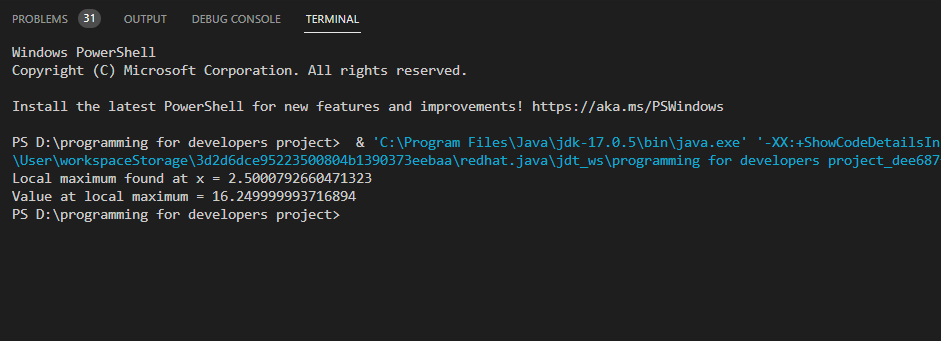
**# Question 5**

1. Implement hill climbing algorithm

**Code;**



**Output;**



**5.b**

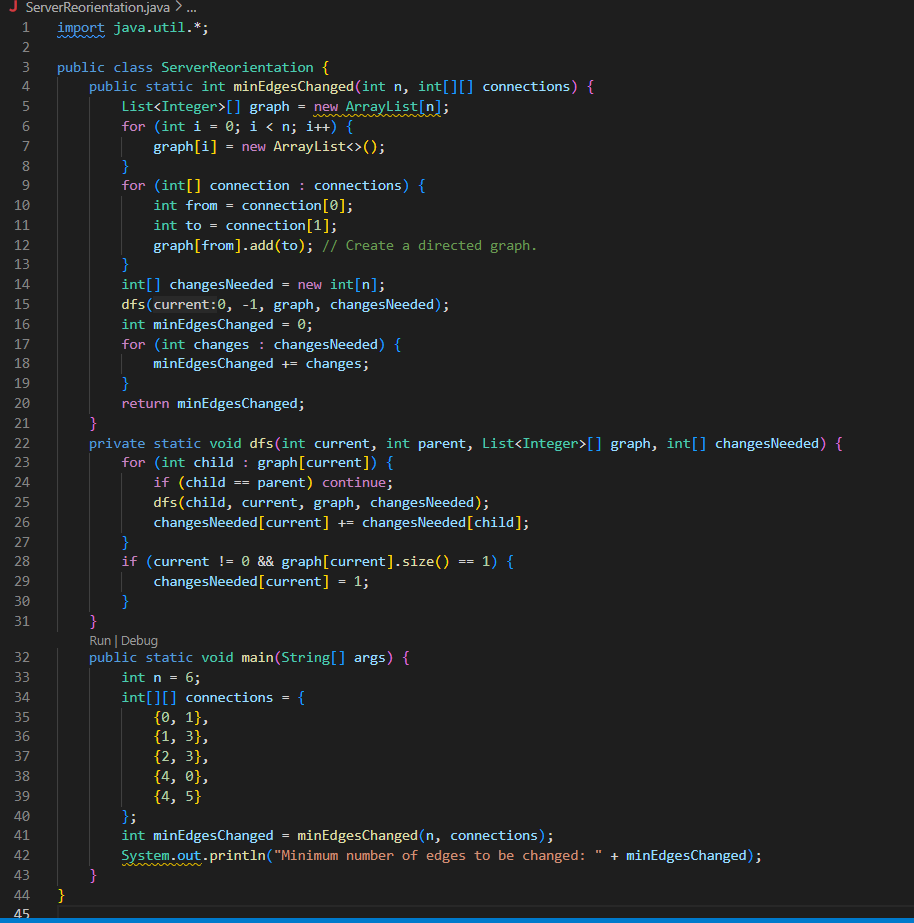
A network consisting of n servers is connected in a tree structure, where the servers are numbered from 0 to n - 1 and there are n - 1 connections between them that only allow for one-way communication. A 2D array a is used to represent these connections, where a[i] = [ai, bi] represents a directed path from server ai to server bi. However, due to specific requirements, all traffic from each server must route to server 0. The task is to reorient some connections to ensure that each server has a path to server 0. The goal is to minimize the number of edges that need to be changed. It is guaranteed that every server must have a path to server 0 after the connections are reordered.

Input: n = 6, connections = [[0,1],[1,3],[2,3],[4,0],[4,5]]

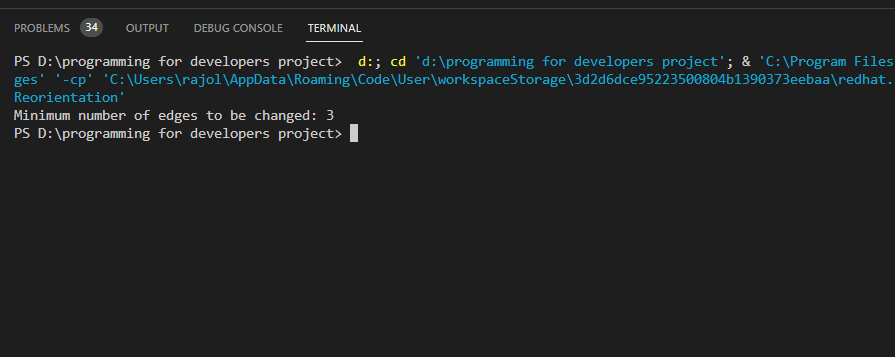
Output: 3

Explanation: Change the direction of edges show in red such that each node can reach the node 0.

**Code;**



**Output;**



**# Question 6**

Write a Java program that uses multithreading to implement a parallel merge sort algorithm. Your program should meet the following requirements:

1. Input: Your program should accept an array of integers as input.

2. Output: Your program should output the sorted array.

3. Algorithm: Your program should use a parallel merge sort algorithm to sort the input array. The algorithm should divide the input array into subarrays, sort the subarrays in parallel using multiple threads, and then merge the sorted subarrays to produce the final sorted array.

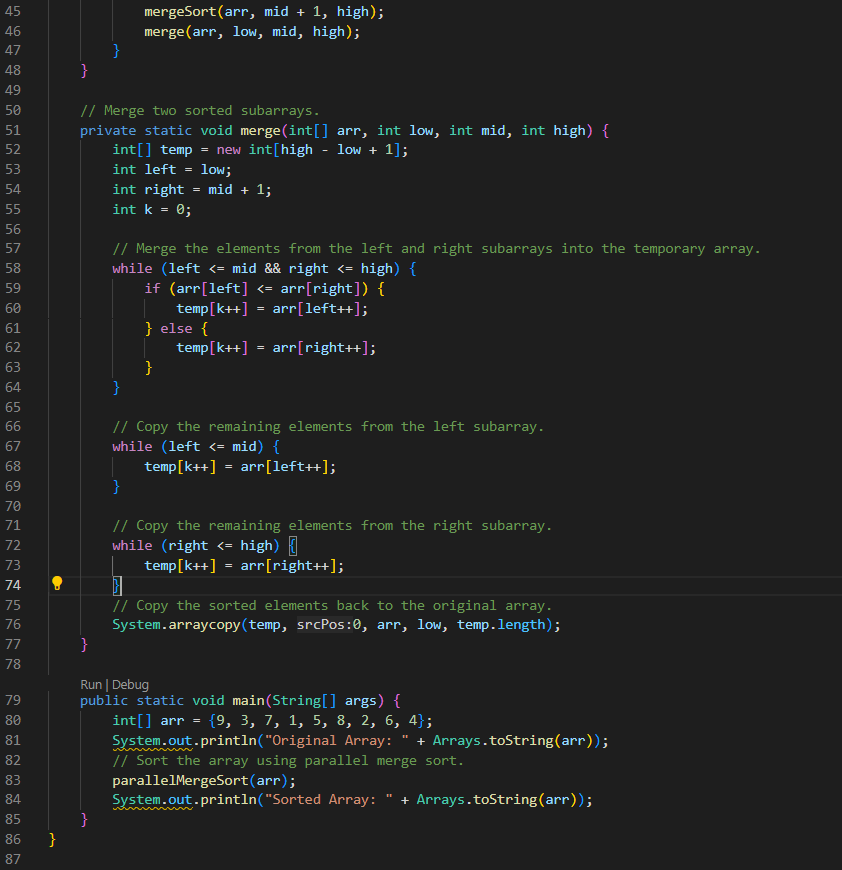
4. Performance: Your program should be optimized for performance, such that it sorts the input array as quickly as possible. You should experiment with different thread counts and input array sizes to find the optimal settings.

5. Error handling: Your program should handle errors and exceptions gracefully, such as by providing informative error messages and exiting gracefully.

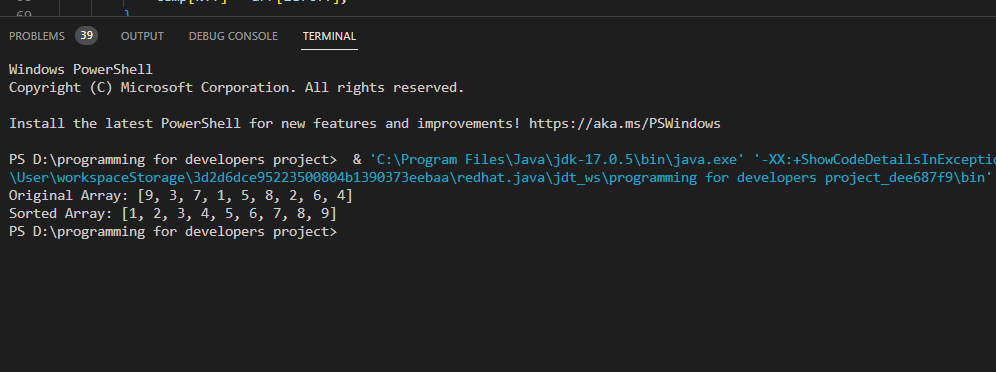
6. Testing: Your program should be thoroughly tested to ensure that it correctly sorts the input array and produces the expected output. To complete this assignment, you will need to implement the parallel merge sort algorithm in Java using multithreading. You should also experiment with different thread counts and input array sizes to find the optimal settings for performance. You can use Java's built-in threading and synchronization features, such as the Thread class and synchronized keyword, to implement the parallel merge sort algorithm.

**Code;**





**Output;**



**#Question 7**

Assignment Title: Social Network Graph Task: Create a GUI application that allows users to visualize a social network graph. Scenario: You have been hired by a social media company to create a tool that visualizes the connections between users. The company wants to see how users are connected and which users have the most influence over others.

Requirements:

1. The application should have a window with a canvas where the graph will be drawn.

2. The nodes of the graph should represent users, and the edges should represent connections between users.

3. The application should read the user data from a file and create the graph accordingly.

4. Each node should display the user's name, profile picture, and the number of followers they have.

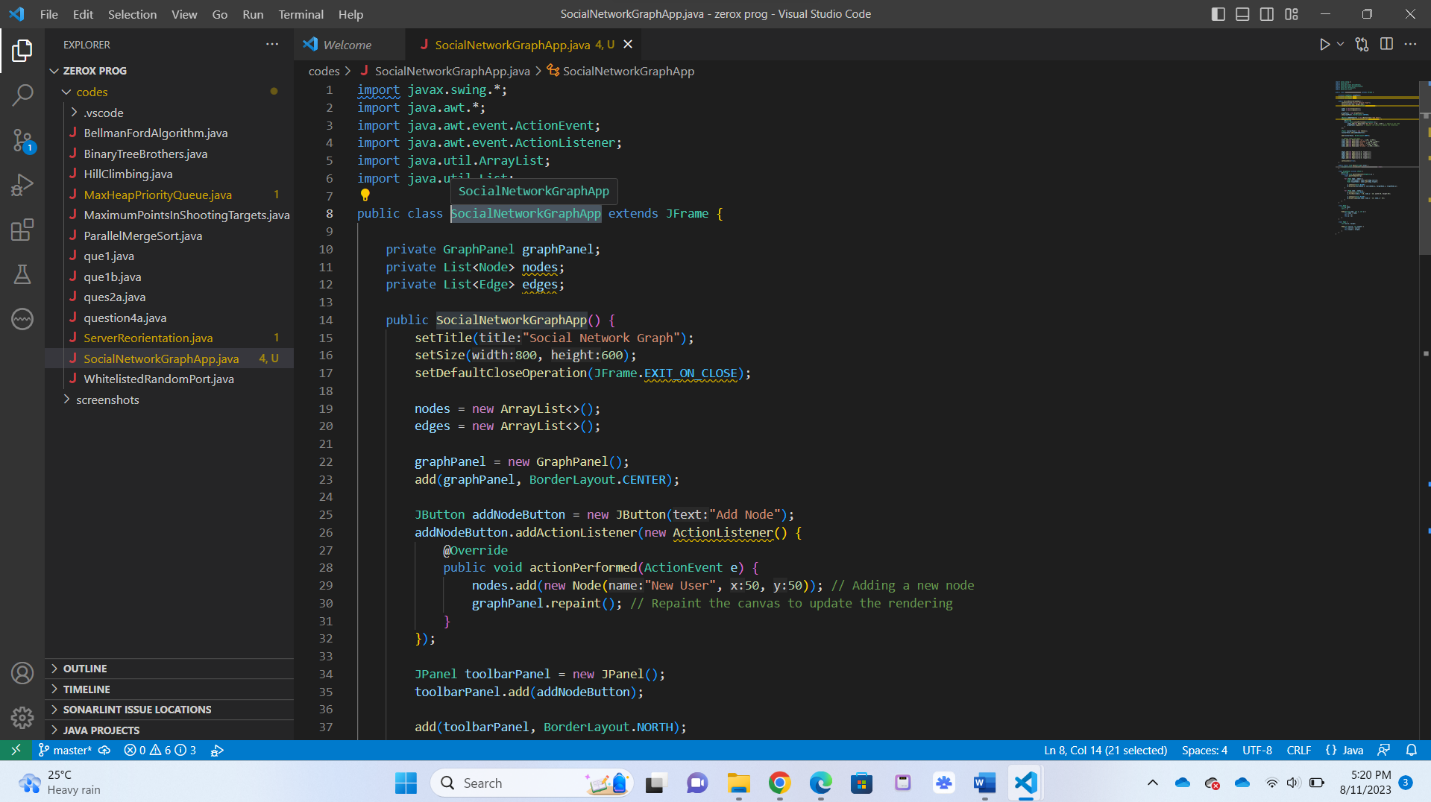
5. The edges should display the strength of the connection between the users, such as the number of likes, comments, or shares between them.

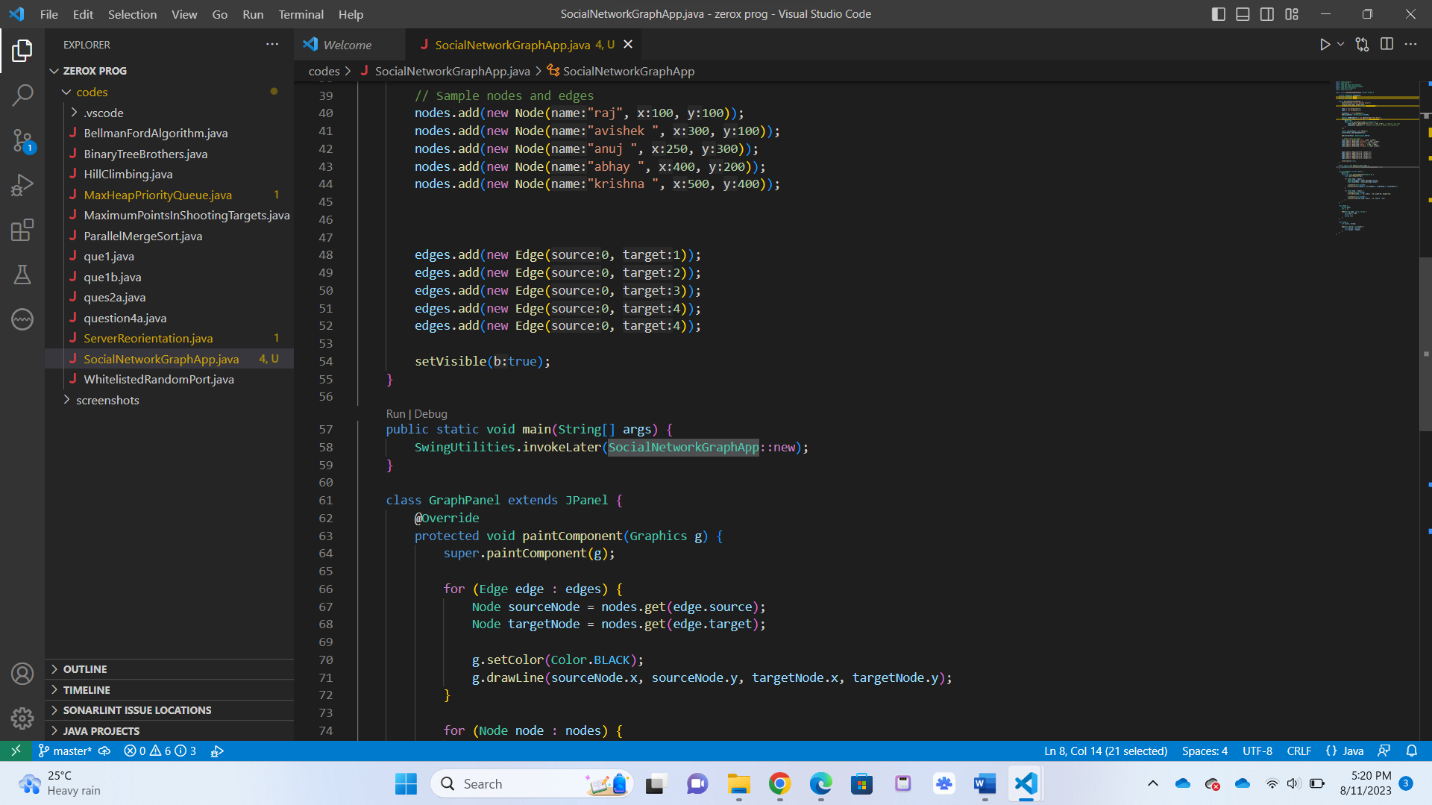
6. The user should be able to select and move nodes around the canvas.

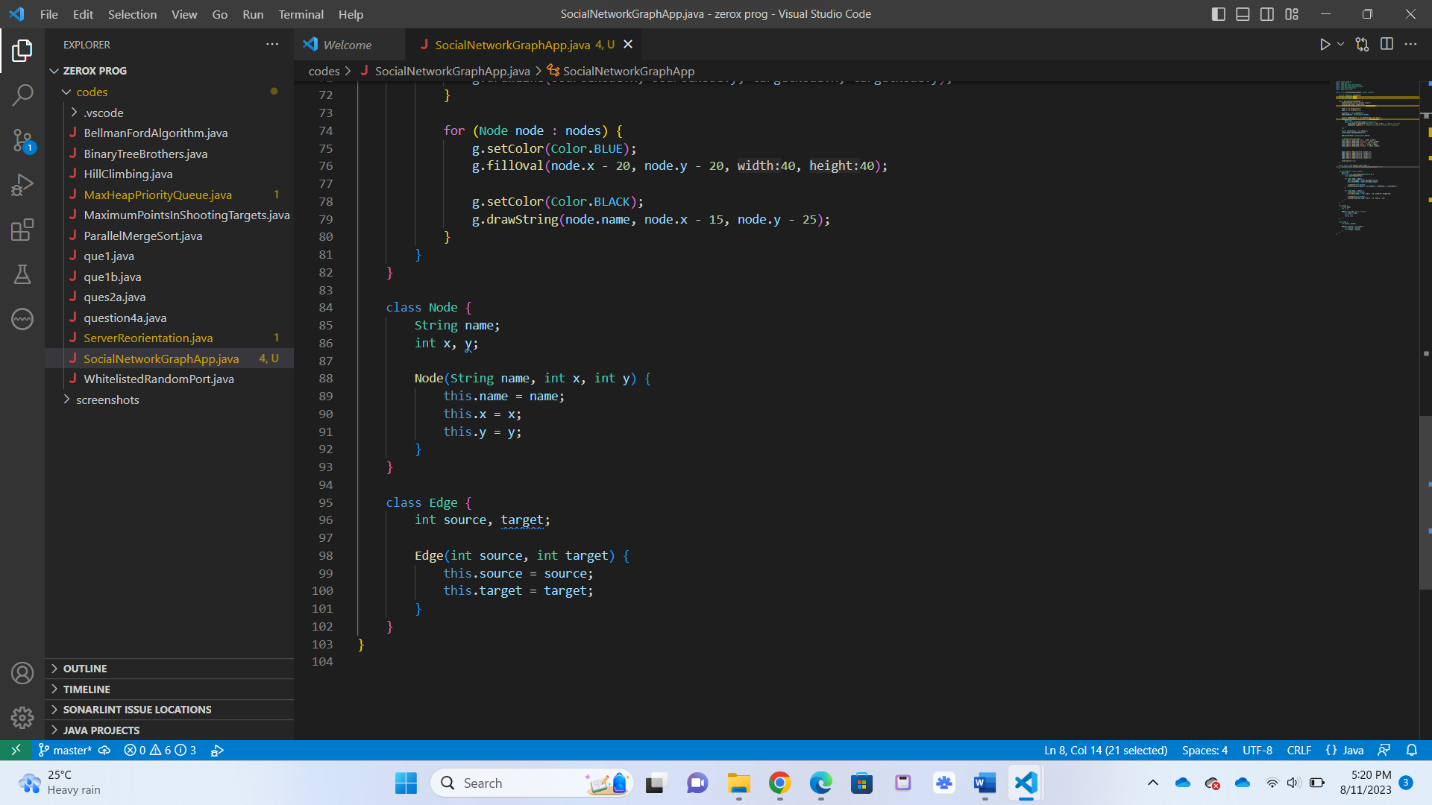
7. The user should be able to delete nodes and edges by selecting them and pressing the delete key.

8. The application should have a toolbar with buttons for selecting mode, adding nodes, and adding edges.

9. The application should allow the user to search for a user and highlight their node and connections.

**Code;**





Output:

