MEMBERS:
- Maria Alejandra Mantilla Coral - A00395792
- Rafaela Sofía Ruiz Pizarro - A00395368
- Andrés David Parra García - A00395676

INTEGRATIVE PROJECT II - Shortest path

You have been hired by a game company to develop a program that allows you to find the shortest path in a maze. The maze is represented by a matrix of size NxM, where each cell in the matrix represents the weight or cost of passing through that cell. A negative value indicates that it is not possible to pass through that cell. You are given a list of different starting positions (or maze entrances). The goal is to find all of the entrances which makes it possible to reach a final position (r,c) and show the one with the minimum total cost in case there is at least one solution, otherwise, it will show “-1”.

Technical specifications:

The program must receive the following data as input:

- Two integers N and M, indicating the number of rows and columns in the matrix.
- An array of size NxM, where each cell in the array represents the weight or cost of passing through that cell. A negative value indicates that it is not possible to pass through that cell.
- An integer P, indicating the number of entrances or starting points the maze has.
- P pairs of integers which will represent each entrance or starting point (ri,ci) (Zero indexed).
- Two more integers r, c indicating the coordinates of the end position to be reached.
- The program should print as output the number of entrances which have a solution, the path with the minimum total cost to reach the end position and the cost of it. If there is no path from the start position to the end position, or if it is not possible to reach the end position from the start position, the program should print "-1".

To find the shortest path with the minimum total cost you need to use algorithms such as Dijkstra's algorithm and to determine if it is possible to reach a position from another one, BFS algorithm. The program must implement both of these algorithms.

Input example:

```
4 4
1   2  4  2
3  -1  2  1
-1 -1  3  8
2  -1  1  0
3
1 0
0 3
3 0
3 2
```

Example output:

```
2
(0,3) -> (1,3) -> (1,2) -> (2,2) -> (3,2)
7
```

Explanation:

- The shortest path from the starting point (1,0) is:
  (1,0) -> (0,0) -> (0,1) -> (0,2) -> (1,2) -> (2,2) -> (3,2)
  With a cost of:
  0 -> +1 -> +2 -> +4 -> +2 -> +3 -> +1 = 13

- The shortest path from the starting point (0,3) is:
  (0,3) -> (1,3) -> (1,2) -> (2,2) -> (3,2)
  With a cost of:
  0 -> +1 -> +2 -> +3 -> +1 = 7

- There is no possible path from the starting point (3,0), since it is surrounded by negative values.

It leaves only two (2) usable entrances and a shortest path of: (3,0) -> (3,1) -> (2,1) -> (2,2) -> (3,2) with a cost of 7.

Note that the cost of the initial node/position is always zero (0) since it takes zero (0) to reach itself.

SOFTWARE ENGINEERING PROBLEM SPECIFICATION TABLE

Client	Gaming company
User	Players
Functional Requirements	**R1:** Register the matrix representing the maze, including its dimensions NxM and the values of each cell. **R2:** Register the coordinates of the initial and final positions. **R3:** Convert the input matrix into a graph. **R4:** Find the shortest path in the graph from an initial position to the final position. **R5:** Display the shortest path from a starting point to the final position.

	R6: Validate if it is possible to access from a position to another into the maze. **R7:** Validate that the input data is correct and within the limits.
Problem Context	The gaming company needs to develop a program that allows players to find the shortest path in a maze represented by an NxM matrix, where each cell in the matrix can have a value of 0 or 1, where 0 indicates an open cell and 1 indicates a blocked cell. The goal is to find the shortest path from an initial position to a final position.
Non-Functional Requirements	**NF1: Efficiency.** The program must be capable of processing and finding the shortest path in the input matrix quickly and efficiently, even for large matrices. **NF2: Accuracy.** The program must be able to find the shortest path in the input matrix accurately and without errors. **NF3: Maintainability.** The program's code must be well-documented, easy to understand, and maintainable by future developers.

FUNCTIONAL REQUIREMENTS ANALYSIS TABLES

Name or identifier	**R1:** Register the matrix representing the maze, including its dimensions NxM and the values of each cell.		
Summary	Allows the user to input a matrix representing the maze to be solved, along with its dimensions NxM. Each cell of the matrix can have a value of 0 or 1, where 0 indicates an open cell and 1 indicates a blocked cell.		
Inputs	Input name	Data type	Selection or repetition condition
	n	int	Only if a negative value is entered.
	m	int	Only if a negative value is entered.

	matrix	int [][]	Only if a value other than 0 or 1 is entered.
General activities necessary to obtain the results	● Read N and M ● Validate that N and M are positive integers. ● Read each cell of the matrix[i][j] ● Validate that each cell contains a value of 0 or 1		
Result or postcondition	The maze is registered by inputting the matrix representation, including its dimensions NxM, and the values of each cell.		
Outputs	Output name	Data type	Selection or repetition condition
	confirmation_msg	String	Only if there was no problem in registering the required data.
	error_msg	String	Only if there was a problem when registering the required data.

Name or identifier	**R2:** Register the coordinates of the initial and final positions.		
Summary	Allows the user to enter the coordinates of the start and end position in the maze.		
Inputs	Input name	Data type	Selection or repetition condition
	r	int	
	c	int	
	p	int	
	ri, ci	int	
General activities necessary to obtain the results	● Read the values of r and c that represent the coordinates of the final position to be reached in the maze. ● Read the value of P indicating the number of entries or starting points of the maze. ● Read the P pairs of integers (ri, ci) that represent the coordinates of each entry or starting point in the maze.		
Result or postcondition	The start and end position coordinates are saved.		
Outpus	Output name	Data type	Selection or repetition condition

	confirmation_msg	String	Only if there was no problem in registering the required data.
	error_msg	String	Only if there was a problem when registering the required data.

Name or identifier	**R3:** Convert the input matrix into a graph.		
Summary	The goal is to convert the input matrix representing the maze into a graph, in order to use pathfinding algorithms on graphs and find the shortest path from the initial position to the final position.		
Inputs	Input name	Data type	Selection or repetition condition
General activities necessary to obtain the results	● Create an empty graph with N x M nodes, one for each cell of the matrix. ● Loop through the array and for each cell that has a positive value: to. Get the index of the corresponding node in the graph. b. Add the edges corresponding to the adjacent nodes in the graph, if they also have a positive value.		
Result or postcondition	The input matrix has been correctly converted into a graph, where the nodes represent the cells of the maze and the edges represent the connections between adjacent cells with no negative values.		
Outputs	Output name	Data type	Selection or repetition condition
	graph	Graph	Only if the graph was successfully generated from the input matrix.
	error_msg	String	Only if there were any problems during the conversion process.

Name or identifier	**R4:** Validate if it is possible to access from a position to another into the maze.		
Summary	The program checks if it is possible to access from an initial position to a final position within the maze, considering the constraints of the maze cells and the existence of a valid path.		
Inputs	Input name	Data type	Selection or repetition condition
General activities necessary to obtain the results	● Verify that the coordinates of the start and end position are within the boundaries of the maze. ● Use a search algorithm, such as the BFS algorithm, to determine if there is a valid path from the start position to the end position in the maze.		
Result or postcondition	It is determined whether it is possible to access from the initial position to the final position within the maze.		
Outputs	Output name	Datatype	Selection or repetition condition
	hasPath	int	

Name or identifier	**R5:** Find and display the shortest path in the graph from an initial position to the final position.		
Summary	The objective is to find the shortest path in the graph generated from the input matrix, from the initial position to the final position. To achieve this we will use the dijkstra algorithm.		
Inputs	Input name	Data type	Selection or repetition condition
	ri	int	
	ci	int	
	rf	int	
	cf	int	
General activities necessary to obtain the results	● Implement Dijkstra's algorithm to find the shortest path from the initial position to the final position in the graph. ● Calculate the total cost of the shortest path found.		
Result or postcondition	The shortest path in the graph from the initial position to the final position is found.		
Outputs	Output name	Datatype	Selection or repetition condition

	shortest_path	String	Only if there is a path from the start position to the end position.
	length_not_found	int	Only if there is no path from the start position to the end position, it will return a -1.
	total_cost_shortest_path	int	

Name or identifier	**R6:** Validate that the input data is correct and within the limits.		
Summary	It verifies that the input data is valid and within the allowed limits, in order to guarantee the correct execution of the program and avoid errors.		
Inputs	Input name	Datatype	Selection or repetition condition
	n	int	
	m	int	
	matrix	int [][]	
	r	int	
	c	int	
	p	int	
	ri, ci	int	
General activities necessary to obtain the results	● Verify that N and M are positive integers. ● Verify that the matrix of size NxM has numeric values. ● Verify that the number of entries P is a positive integer. ● Verify that each pair of input coordinates (ri, ci) is within the bounds of the matrix (0 ≤ ri < N, 0 ≤ ci < M). ● Verify that the end position coordinates (r, c) are within the matrix bounds (0 ≤ r < N, 0 ≤ c < M).		
Result or postcondition	If all the input data is valid and within the allowed limits, no message is displayed. Otherwise, an error message will be displayed.		
Outputs	Output name	Datatype	Selection or repetition condition
	error_msg	String	Only if any of the input data is invalid.