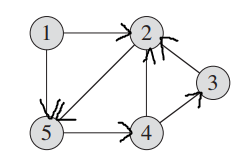
**BFS adjacency list:**

****

insertNode(int position, V v)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 6 | void | void | void | The size of the adjacency list increases by 1 |
| V | 6 | void | void | void |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 1 | void | void | void | Message : “Node already exists”. As shown in the graph. |
| V | 1 | void | void | void |

insertEdge(int position1, int position2)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 2 | void | void | void | Edge added to the graph |
| Int | 1 | void | void | void |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 1 | void | void | void | Message : “The Node does not exist”. As shown in the graph. |
| Int | 8 | void | void | void |

BFS(int position)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 1 | String[] | 1 2 3 4 5 | 1 2 3 4 5 | All the Nodes from the starting Node were visited. Being a connected graph from that Node. |

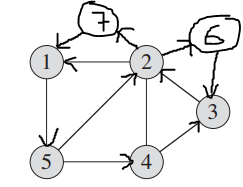
| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 3 | String[] | 1 2 3 4 5 | 2 3 4 5 | There was a missing Node. So the graph is not really a Strongly connected graph |

**BFS matrix:**

BFSMatrixAdjacencyList(int[][] matrix, int position)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| int [][] | {{0, 1, 0, 0, 1, 0},  {0, 0, 0, 0, 1, 0},  {0, 1, 0, 0, 0, 0},  {0, 1, 1, 0, 0, 0},  {0, 0, 0, 1, 0, 0},  {0, 0, 0, 0, 0, 0} | String [] | 0 1 4 3 2 5 | 0 1 4 3 2 null l | There is a node that has no connections, that's why there is a null in the array obtained. |
| int | 0 | String [] | 0 1 4 3 2 5 | 0 1 4 3 2 null |
| int [][] | {{0, 1, 0, 0, 1},  {0, 0, 0, 0, 1},  {0, 1, 0, 0, 0},  {0, 1, 1, 0, 0},  {0, 0, 0, 1, 0}, | String [] | 0 1 4 3 2 | 0 1 4 3 2 | All nodes are connected between each other that’s why the output is the expected. |
| int | 0 | String [] | 0 1 4 3 2 | 0 1 4 3 2 |

**DFS adjacency list:**

****

insertNode(int position, V v)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 1 | void | void | void | Message : “Node already exists”. As shown in the graph. |
| V | 1 | void | void | void |
| Int | 2 | void | void | void | Message : “Node already exists”. As shown in the graph. |
| V | 2 | void | void | void |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 6 | void | void | void | The size of the adjacency list increases by 1 |
| V | 6 | void | void | void |
| Int | 0 | void | void | void | The size of the adjacency list increases by 1 |
| V | 0 | void | void | void |
| Int | 0 | void | void | void | Message : “Node already exists”. As shown in the graph. |
| V | 0 | void | void | void |

insertEdge(int position1, int position2)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 1 | void | void | void | Edge added to the graph |
| Int | 1 | void | void | void |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 1 | void | void | void | Edge added to the graph |
| Int | 2 | void | void | void |
| Int | 2 | void | void | void | Edge added to the graph |
| Int | 1 | void | void | void |

DFS(int position)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 3 | void | void | void | All the Nodes from the starting Node were visited. Being a connected graph from that Node. |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 7 | void | void | void | All the Nodes from the starting Node were visited. Being a connected graph from that Node. |

DFSVisit(Node<V> node)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Node<V> | (1, 1) | void | void | void | Node conected to a previous one |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Node<V> | (2, 2) | void | void | void | Node conected to a previous one |

**DFS matrix:**

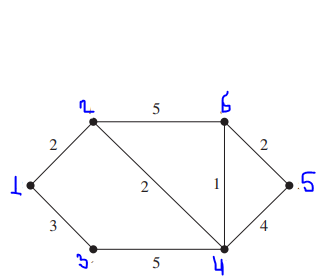
DFS(int[][] matrix, int position)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| matrix[0][0] | 1 | String | 2  3  4  1 | 2  3  4  1 | The result is the expected because the graph is altamente conexo |
| int | 0 |
| matrix[2][3] | 1 | String | 2  3  4  1 | 2  3  4  1 | The result is the expected because the graph is altamente conexo |
| int | 0 |

DFSVisit(int[][] matrix, int position, int[] color, int[] father, int[] distance)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| int[[][] | [values] | int | it depends | it depends | Is very difficult to make a test case for this method because it’s recursive and the output is different depending on the matrix given. |
| int | it depends |
| int[] | [values] |
| int[][] | [values] |
| int[][] | [values] |

**DIJKSTRA adjacency list:**

****

Dijkstra(int V)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 7 | void | void | void | The size of the adjacency list increases by 1 |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 7 | void | void | void | The size of the adjacency list increases by 1 |
| Int | 3 | void | void | void | The size of the adjacency list increases by 1 |

addEdge(int u, int v, int w)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 3 | void | void | void | Edge added to the graph |
| Int | 6 | void | void | void |
| Int | 2 | void | void | void |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 3 | void | void | void | Edge added to the graph |
| Int | 2 | void | void | void |
| Int | 3 | void | void | void |
| Int | 2 | void | void | void | Edge added to the graph |
| Int | 3 | void | void | void |
| Int | 2 | void | void | void |

dijkstra(int s)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 3 | void | void | void | Finds the shortest path from this Node to the end Node |

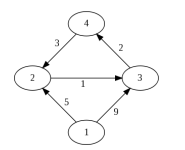
| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 8 | void | void | void | Message : “No path from 8 to (End Node)”. Due to the nonexistent Node 8. |

**DIJKSTRA matrix:**

dijkstra(Heap<Integer> heap, int[][] matrix, int source, int destination)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| heap[integer] | null | String | The shortest path between 0 and 5 is:  0 1 3 5 The distance is: 5 | The shortest path between 0 and 5 is:  0 1 3 5 The distance is: 5 | The shortest path continues to be the expected because the new vertex added is very heavy. |
| int[][] | [0][5] = 6 |
| int | 0 |
| int | 5 |
| heap[integer] | null | String | The shortest path between 0 and 5 is:  0 1 3 5 The distance is: 5 | The shortest path between 0 and 5 is:  0 5 The distance is: 2 | The shortest path and it’s weight changed because of the new vertex added. |
| int[][] | [0][5] = 2 |
| int | 0 |
| int | 5 |

**FLOYD-WARSHALL adjacency list:**

****

FloydWarshall(int V)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 5 | void | void | void | The size of the adjacency list increases by 1 |
| Int | 1 | void | void | void | The size of the adjacency list increases by 1 |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 10 | void | void | void | The size of the adjacency list increases by 1 |
| Int | 20 | void | void | void | The size of the adjacency list increases by 1 |

addEdge(int u, int v, int w)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 4 | void | void | void | Edge added to the graph |
| Int | 1 | void | void | void |
| Int | 5 | void | void | void |
| Int | 1 | void | void | void | Edge added to the graph |
| Int | 3 | void | void | void |
| Int | 2 | void | void | void |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 4 | void | void | void | Node 8 is nonexistent. So no Edge added |
| Int | 8 | void | void | void |
| Int | 5 | void | void | void |
| Int | 1 | void | void | void | Node 7 is nonexistent. So no Edge added |
| Int | 7 | void | void | void |
| int | 10 | void | void | void |

floydWarshall()

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| void | void | void | void | void | Obtains the shortest path from the Initial Node to the End Node. As well as its distance. |

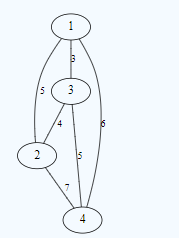
| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| void | void | void | void | void | Obtains the shortest path from the start of the graph to its end. As well as its distance. |

**FLOYD-WARSHALL matrix:**

floyd(int[][] graph)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| int[][] | [2][1] = 6 | void | void | void | A new vertex was added and the shortest distance between 1 4 is the same |
| int [][] | [1][4] =7 | void | void | void | A new vertex was added and the shortest distance between 1 4 is now 7 |

**KRUSKAL adjacency list:**

****

createSet(int gNodes)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 4 | void | void | void | Create a set for each node of the list |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 10 | void | void | void | Create a set for each node of the list |

find(int x)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 2 | Int | 2 | 2 | It searches for the set an int is stored in, and returns the set |
| Int | 3 | Int | 3 | 3 | It searches for the set an int is stored in, and returns the set |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 5 | void | void | void | There is no Node 5, therefore no set for 5. |

merge(int u, int v)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 1 | void | void | void | Change the set of the lower value node for the set of the greater value Node |
| Int | 2 | void | void | void |

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| Int | 4 | void | void | void | Change the set of the lower value node for the set of the greater value Node |
| Int | 1 | void | void | void |
| Int | 6 | void | void | void | There is no Edge or Node 6. Therefore, the set of Node 1 is not altered |
| Int | 1 | void | void | void |

bubbleSort(ArrayList<String[]> Edges)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| ArrayList<String[]> | Edges | void | void | void | The array of Edges is organized by its weight from lowest to greatest |

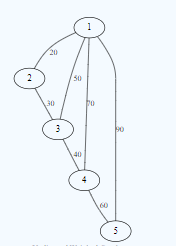
| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| ArrayList<String[]> | Weights | void | void | void | Error. It is needed to know the Node of the edge starts in and the Node of the edge ends in. |

**KRUSKAL matrix:**

kruskals(int[][] g)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| int [][] | [1][2] = 1 | int | 6 | 7 | The weight between the nodes 1 and 2 changed, and the value obtained too. |
| int [] [] | [1][4] | int | error | error | There are only 4 nodes and its impossible to add a connection to a node that doesn't exist |

**PRIM adjacency list:**

****

prim(Heap<Integer> heap, ArrayList<ArrayList<Integer>> adjList, ArrayList<ArrayList<Integer>> adjListWeight, int start)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| heap | null | String | 0 - 1 : 20  1 - 2 : 30  0 - 3 : 70  3 - 4 : 60 | 0 - 1 : 20  1 - 2 : 30  0 - 3 : 70  3 - 4 : 60 | The tree continues to be the same because the weight did not affect the result |
| adj | (1,2) |
| adjWeight | 9 |
| int | 0 |

**PRIM matrix:**

prim(Heap<V> heap, int[][] graph, int start)

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| heap | null | String | 0 - 1 : 20  1 - 2 : 30  0 - 3 : 70  3 - 4 : 60 | 0 - 1 : 200  1 - 2 : 30  0 - 3 : 70  3 - 4 : 60 | The weight between 1 and 0 was over righted and that's why the result changed |
| int[][] | [1][0] = 200 |
| int | 0 |

**SOLUTION OF THE PROBLEM:**

The final version of the program posses different ins and outs, as well as a randomizer. That's why for each test case, we are going to create different tables for the different moments in the program.

For this 1 test case, we verify that the game ends getting the correct results of all the equations.

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| String | Luispi |  |  |  | This moment corresponds to the username setup |
|  |  | String | Randomizer | 40\*10 | This moment corresponds to the mathematical equation found on the first room/node |
| Int | 400 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | Randomizer | 163+457 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 620 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | Randomizer | 800-69 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 731 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | Randomizer | 424/188 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 2 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct (rounded to the nearest whole number), the user advances to the next node of the path |
|  |  | String | Randomizer | 859-509 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 350 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | Randomizer | 294/175 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 2 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct (rounded to the nearest whole number), the user advances to the next node of the path |
|  |  | String | Randomizer | 999\*136 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 135864 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | Randomizer | 53-666 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | -613 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | Randomizer | 579+500 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 1079 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | Randomizer | 965\*575 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 554875 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | Randomizer | 561+451 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 1012 | String | message : “Well done” | message : “Well done” | Correspond to the answer of the previous equation, and it’s verification. As it was correct, the user advances to the next node of the path |
|  |  | String | message : “Yo have stopped Mathlactus” | message : “Yo have stopped Mathlactus” | End of the game |

For this 2 test case, we verify that the game ends getting the incorrect results of all the equations.

| Entradas | | Salidas | | | Observaciones |
| --- | --- | --- | --- | --- | --- |
| Dato | Valor | Dato | Vlr Esperado | Vlr obtenido |  |
| String | Luispi |  |  |  | This moment corresponds to the username setup |
|  |  | String | Randomizer | 40\*10 | This moment corresponds to the mathematical equation found on the first room/node |
| Int | 10 | String | message : “Incorrect” | message : “Incorrect” | Correspond to the answer of the previous equation, and it’s verification. As it was incorrect, the user passes to the next node of the continuous path |
|  |  | String | Randomizer | 163+457 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 0 | String | message : “Incorrect” | message : “Incorrect” | Correspond to the answer of the previous equation, and it’s verification. As it was incorrect, the user passes to the next node of the continuous path |
|  |  | String | Randomizer | 800-69 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 100 | String | message : “Incorrect” | message : “Incorrect” | Correspond to the answer of the previous equation, and it’s verification. As it was incorrect, the user passes to the next node of the continuous path |
|  |  | String | Randomizer | 424/188 | This moment corresponds to the mathematical equation found on the next room/node |
| Int | 100000 | String | message : “Incorrect” | message : “Incorrect” | Correspond to the answer of the previous equation, and it’s verification. As it was incorrect, the user passes to the next node of the continuous path |
|  |  | String | message : “GameOver” | message : “GameOver” | End of the game |

**Group members:**

**A00381777 | Belalcazar Bonelo, Santiago Jose**

**A00381157 | Escobar Vacaflor, Daniel**

**A00381323 | Pinillos Sanchez, Luis Fernando**