Unit Cases Tests

111 ∞	112 ∞	113 ∞	114 ∞	115 ∞	116 21	117 ∞	118 ∞	119 ∞	120 ∞	121 ∞
100 ∞	101 ∞	102 ∞	103 ∞	104 ∞	105 20	106 ∞	107 ∞	108 ∞	109 ∞	110 ∞
89 ∞	90 ∞	91 ∞	92 ∞	93 18	94 19	95 ∞	96 ∞	97 ∞	98 ∞	99 ∞
78 ∞	79 ∞	80 ∞	81 ∞	82 17	X	84 ∞	85 ∞	86 ∞	87 ∞	88 ∞
67 ∞	68 ∞	69 ∞	70 ∞	71 16	72 15	73 14	74 13	75 ∞	76 ∞	77 ∞
56 ∞	57 ∞	58 ∞	X	X	X	X	63 12	X	65 ∞	66 ∞
45 ∞	46 ∞	47 ∞	48 7	49 8	50 9	51 10	52 11	53 ∞	54 ∞	55 ∞
34 ∞	35 ∞	36 ∞	37 6	X	X	40 ∞	X	42 ∞	43 ∞	44 ∞
23 ∞	24 ∞	25 ∞	26 5	27 4	28 3	29 ∞	30 ∞	31 ∞	32 ∞	33 ∞
12 ∞	13 ∞	14 ∞	15 ∞	16 ∞	17 2	18 ∞	19 ∞	20 ∞	21 ∞	22 ∞
1 ∞	2 ∞	3 ∞	4 ∞	5 ∞	6 1	7 ∞	8 ∞	9 ∞	10 ∞	11 ∞

UnitCasesTest

Scenes:

Name	**Class**	**Scenery**
testScene1	**GraphA**	**Create a graph A with 5 vertex**
testScene2	**GraphA**	**Create a graphA with 5 vertex and link them with edges**

testScene3	**GraphA**	**Create a graphA and link them in two branches**

testScene1	**GraphB**	**Create a graph B with 5 vertex**
testScene2	**GraphB**	**Create a graphB with 5 vertex and link them with edges**
testScene3	**GraphB**	**Create a graphB and link them in two branches**
gameScene1	**Game**	**Create a Game and link the matrix**

GraphA

aim: Verify if the graph's edges were created				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphA	**addOneEdgeTest**	**testScene1**	**A unlinked Graph**	**A linked graph**

aim: Verify if the graph's edges were created				
Class	**Method**	**Scenery**	**Input**	**Output**

GraphA	**addMultipleEdgeTest**	**testScene1**	**A unlinked Graph**	**A linked graph**

aim: Verify the distance between vertex				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphA	**DijkstraTestToNearestVertex**	**testScene2**	**A linked Graph**	**the shortest distance between a origin vertex and his nearest linked vertex**

aim: Verify the distance between vertex				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphA	**DijkstraTestToFarthestVertex**	**testScene2**	**A linked Graph**	**the shortest distance between a origin vertex and his farthest linked vertex**

aim: Found the smallest edge linked vertex

Class	Method	Scenery	Input	Output
GraphA	primSmallerEdge	testScene2	A linked Graph	Found the smallest edge’s link vertex

aim: Found the last edge linked vertex				
Class	Method	Scenery	Input	Output
GraphA	primGreaterEdge	testScene2	A linked Graph	Found the last edge’s link vertex

aim: Found the minor edge linked vertex				
Class	Method	Scenery	Input	Output
GraphA	KruskalMinorEdge	testScene2	A linked Graph	Found the minor edge’s link vertex

aim: Found the greatest edge linked vertex				
Class	Method	Scenery	Input	Output
GraphA	KruskalGreatestEdge	testScene2	A linked Graph	Found the greatest edge’s link vertex

aim: Found the matrix edge’s position				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphA	**FloydWarshallTest**	**testScene2**	**A linked Graph**	**Found the matrix edge’s position**

aim: Found and create trees using the linked graphs				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphA	**DFSForestTest**	**testScene3**	**A linked Graph**	**Linked graphs trees**

aim: Found the vertex´s son				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphA	**BFSTest**	**testScene3**	**A linked Graph**	**The linked vertex required**

Graph B

aim: Verify if the graph’s edges were created				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphB	**addOneEdgeTest**	**testScene1**	**A unlinked Graph**	**A linked graph**

aim: Verify if the graph’s edges were created				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphB	**addMultipleEdgeTest**	**testScene1**	**A unlinked Graph**	**A linked graph**

aim: Verify the distance between vertex				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphB	**DijkstraTestToNearestVertex**	**testScene2**	**A linked Graph**	**the shortest distance between a origin vertex and his nearest linked vertex**

aim: Verify the distance between vertex				
Class	**Method**	**Scenery**	**Input**	**Output**

GraphB	**DijkstraTestToFarthestVertex**	**testScene2**	**A linked Graph**	**the shortest distance between a origin vertex and his farthest linked vertex**

aim: Found the smallest edge linked vertex				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphB	**primSmallerEdge**	**testScene2**	**A linked Graph**	**Found the smallest edge's link vertex**

aim: Found the last edge linked vertex				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphA	**primGreaterEdge**	**testScene2**	**A linked Graph**	**Found the last edge's link vertex**

aim: Found the minor edge linked vertex				
Class	**Method**	**Scenery**	**Input**	**Output**

GraphB	**KruskalMinorEdge**	**testScene2**	**A linked Graph**	**Found the minor edge's link vertex**

aim: Found the greatest edge linked vertex				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphB	**KruskalGreatestEdge**	**testScene2**	**A linked Graph**	**Found the greatest edge's link vertex**

aim: Found the matrix edge's position				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphB	**FloydWarshallTest**	**testScene2**	**A linked Graph**	**Found the matrix edge's position**

aim: Found and create trees using the linked graphs				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphB	**DFSForestTest**	**testScene3**	**A linked Graph**	**Linked graphs trees**

aim: Found the vertex´s son				
Class	**Method**	**Scenery**	**Input**	**Output**
GraphB	**BFSTest**	**testScene3**	**A linked Graph**	**The linked vertex required**

Game:

aim: Use the BFS to find the best path				
Class	**Method**	**Scenery**	**Input**	**Output**
Game	**giveBFSTest**	**gameScene1**	**A linked board**	**The best path**

aim: Use the Dijkstra to find the best path				
Class	**Method**	**Scenery**	**Input**	**Output**
Game	**giveDijkstraTest**	**gameScene1**	**A linked board**	**The best path**

aim: Generate a random value for each vertex in the graph				
Class	**Method**	**Scenery**	**Input**	**Output**

Game	**generateRandomsTest**	**gameScene1**	**A linked board**	**Values for each vertex**

aim: Link all the vertex in the board and give them a value				
Class	**Method**	**Scenery**	**Input**	**Output**
Game	**linkMatrixTest**	**gameScene1**	**123 vertex**	**A linked board**

aim: Generate a generic obstacle				
Class	**Method**	**Scenery**	**Input**	**Output**
Game	**genericObstacleTest**	**gameScene1**	**A linked board**	**A generic obstacle**

aim: Verify if the ladderboard is created				
Class	**Method**	**Scenery**	**Input**	**Output**
Game	**finishGameTest**	**gameScene1**	**A player**	**an arraylist with all the winners and their score**