	Summary	Pre Condition	Post condition	Flow of events	Test Design	Expected result	Actual result	Status
Start the program	When the program is launched for the first time it must load the default configuration of block and set the labels at their correct value	The program is not running	The program is running correctly	1) The program is launch 2)The blocks are set to their default configuration 3)The move counter label is set to 0 4)The level label is set to 1	The test checks if: - the identity string of the load configuration is equal to the default configuration one - The variable text of the move counter is equal to 0 - The variable text of the level labels is equal to 1	The expected results are: - The identity string is 24422442233221121001 - The variable text of the move counter is 0 - The variable text of the level label is 1	The actual results are: - The identity string is 24422442233221121001 - The variable text of the move counter is 0 The variable text of the level label is 1	PASS
Move a block	The player selects a block and moves it in a legal position	The block must have some blank spaces adjacent to it so that neither the block overlaps any other block nor it leaves the game board	The block is set in the selected position and it is placed correctly	1) The player left clicks on a block 2) The player releases the left button on the desired end position 3) The selected bloc moves towards the end position 4)The block stops as soon as it finds another block or the board's edges	The test performs a series of moves with different end position, one for critical case: - One is a legal move - One is an out of bound move - One is an overlap move For each move performed, the test checks if the end position is the expected one for that type of move.	The expected results are: - For the legal move, the block reaches the end position - For the out bound move, the block stops at the edge of the board - For the overlap move, the block stops before overlapping the other ones	The actual results are: - For the legal move, the block reaches the end position - For the out bound move, the block stops at the edge of the board For the overlap move, the block stops before overlapping the other ones	PASS
Undo the last move	When the player clicks on the undo button, the last perform move in undone and the move counter is decreased by 1	At least one move has been done	The last block moved returns to its previous position	1) The player clicks on the undo button 2) The last block moved returns to its previous position 3) The move counter is decreased by 1	The test performs a generic move from the default configuration of blocks, then it tries to perform the undo method and checks if the position of the moved block is returns to the start one and if the move counter is decreased by 1	The expected results are: - The last performed move is correctly undone - The move counter is decreased by 1	The actual results are: - The last performed move is correctly undone The move counter is decreased by 1	PASS
Redo the last move	When the player clicks on the redo button, if there is at least one undone move, it will be redone and the move counter is increased by 1	At least one undone move is present	The last undone move is redone and the move counter is increased by 1	1) The player clicks the redo button 2) the last undone move is reversed 3)The move counter is increased by 1	The test performs a generic move from the default blocks configuration, then it executes the undo and redo methods. In the end it checks if the blocks are returned to the positions in which they were before the undo method and checks if the move counter is equals to 1	The expected results are: - The last undo move is reversed and the blocks are return to their before-undo positions - The move counter is equals to 1	The actual results are: - The last undo move is reversed and the blocks are return to their before-undo positions - The move counter is equals to 1	PASS
Best next move	When the player clicks on the hint button, the best next possible move is performed by the system.	The program is running and the connection with the db is correctly open	A block is moved so the level can be completed in the minimum amount of moves	1) The player clicks on the hint button 2) The system asks the db to find the next best move 3) The next best move is performed	The test executes the hint function on the default block configuration and check if the move performed is equal to the one found on an online solver	The expected result is that the board state after the hint function is: "24422442233220121101"	The actual result is that the board state after the hint function is: "24422442233220121101"	PASS
End game	When the large block reaches the bottom mid edge of the board, the game is won	The large block isn't already in the win position	The large block is in the win position and the game is won	1) The player moves the large block in the bottom mid position 2) The win is consider won	The test tries two different situations: - A generic block is moved to the win position - The large block is move to the win position After each move, the test checks if the game is consider won	The expected results are: - When a generic block reaches the win position nothing happens and the game continues normally - When the large block reaches the win position, the game is consider won	The actual results are: - When a generic block reaches the win position nothing happens and the game continues normally - When the large block reaches the win position, the game is consider won	PASS
Quit the game	When the player clicks on the exit button, the game close, without perform any other action	The program is running	The program is not running	1) The player clicks on the exit button 2) The program is closed correctly	The test checks that when the exit button is pressed, the system exit function is called correctly	The expected result is that the exit status of the exit all is equals to 0	The actual result is that the exit status of the exit all is equals to 0	PASS
Save the game	When the player clicks on the save button, the state of the game is converted into a json file, that is saved in a specific position, choose by the user	The program is running	The board is converted to a Json file that is save in a specific directory, choose by the player	1) The player clicks on the save button 2) The board is converted to a Json file 3) The Json file is saved in the chosen directory	The test calls the save method on a default configuration of blocks, then load the file that is just been saved and get the state of the board that is been just loaded.	The expected result is that the state of the load board is equals to the default block configuration state	The actual result is that the state of the load board is equals to the default block configuration state	PASS
Load the game	When the player clicks on the load button, a new window is shown and the player can select a json file to load	The program is running and there is a compatible json file	The blocks configuration is set to the one describes by the json file loaded. The move counter and level label are loaded too	1) The player clicks on the load button 2) The system allows the player to select the file to load 3) The blocks configuration, the move counter and the level label are changed based on the loaded file	The test calls the save method on a default configuration of blocks, then loads the file that is just been saved and gets: the state of the board that is just been loaded, the move counter and the level label. Then checks if the state is the same as the default blocks configuration, if the move counter and the level label are equals to the ones saved in the file	The expected results are: - the state of the load board is equals to the default block configuration state - The move counter and the level label are equals to the saved ones	The actual results are: - the state of the load board is equals to the default block configuration state - The move counter and the level label are equals to the saved ones	PASS
Change the initial blocks layout	When the player clicks on the levels, a new window is shown and the player can select a new block configuration	The program is running and the db connection is open	The blocks configuration of the chosen level is load into the game and the level label shows the level selected	1) The player clicks on the levels button 2) The player selects a new level 3) The new blocks configuration is load 4) The level label is set to the chosen level number and the move counter is set to 0	The test opens a connection with the db and select a specific level (420) different from the default one. Then it checks that the state of the load blocks configurations is the same as the state of the level selected	The expected result is the string: 44124402103312331233. The level label must be set on 420	The actual result is the string: 44124402103312331233. The level label is set on 420	PASS
Start a new game	When the player clicks on the new game button, the board is reset to the last configuration loaded and the move counter is set to 0	The program is running	The board is reset to the last configuration load and the move counter is set 0	1) The player clicks on the new game button 2) The board is reset to the last configuration loaded 3) The move counter is set to 0	The test performs two generic moves starting from the default blocks configuration and then recall the new game feature. Then it checks if the board state has returned to the default one and if the move counter is reset to 0	The expected results are: - The board state after the new game feature is equals to the default one - The move counter is reset to 0	The actual results are: - The board state after the new game feature is equals to the default one The move counter is reset to 0	PASS
Get the state of the board	When called, this function transforms the blocks configuration in a unique string of number that represents that specific board situation.	A board is initialized	The board is represented by a unique string of number	1) The getState method is called 2) The board state is converted to a string of number	The test takes the default blocks configuration and convert it into the identity string, reading the matrix of the board line by line, left to right and converting the block into their representative number	The expected result is the string: 24422442233221121001	The actual result is the string: 24422442233221121001	PASS